ENHANCED INCREMENTAL DYNAMIC COMMUNITY DETECTION FOR IMPROVING THE STABILITY OF COMMUNITY STRUCTURE IN NETWORK ANALYSIS

SITI HARYANTI BINTI HJ HAIROL ANUAR

DOCTOR OF PHILOSOPHY

2025

Faculty of Information and Communications Technology

ENHANCED INCREMENTAL DYNAMIC COMMUNITY DETECTION FOR IMPROVING THE STABILITY OF COMMUNITY STRUCTURE IN NETWORK ANALYSIS

Siti Haryanti binti Hj Hairol Anuar

Doctor of Philosophy

2025

ENHANCED INCREMENTAL DYNAMIC COMMUNITY DETECTION FOR IMPROVING THE STABILITY OF COMMUNITY STRUCTURE IN NETWORK ANALYSIS

SITI HARYANTI BINTI HAIROL ANUAR

A thesis submitted
in fulfillment of the requirements for the degree of
Doctor of Philosophy

Faculty of Information and Communications Technology

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2025

DECLARATION

I declare that this thesis entitled “Enhanced Incremental Dynamic Community Detection for Improving the Stability Community Structure in Network Analysis“ is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature :.....

Name : Siti Haryanti binti Hj Hairol Anuar

Date : 25 November 2025

APPROVAL

I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of Doctor of Philosophy.

Signature :.....

Supervisor's Name : Prof. Madya Ts. Dr. Zuraida binti Abal Abas

Date : 25 November 2025

DEDICATION

In the Name of Allah, the Most Beneficient, the Most Merciful.

I begin this dedication with heartfelt praise and gratitude to Allah SWT for granting me the strength, wisdom, and guidance to complete this academic journey.

To my beloved husband, your unwavering encouragement and sacrifices have kept me focused and determined during the challenges upon this thesis completion.

To my precious children, your presence fills my life with joy and purpose. Your innocent smiles have been a constant source of inspiration, reminding me of countless blessings from Allah.

I am deeply grateful to my parents, as well as my parents-in-law, my siblings, extended family and friends, for the prayers, support, and encouragement given to me throughout this journey.

Lastly, I dedicate this work to the Prophet Muhammad (peace be upon him), whose teachings and example continue to illuminate our paths. May this thesis be a reflection of seeking knowledge and understanding, as promoted by Islam religion. May Allah accept this effort and make it useful for all seeking the knowledge and truth.

Praises to Allah, the One and Only.

ABSTRACT

Network analysis plays a crucial role in detecting communities within complex networks, which are prevalent across diverse domains and exhibit intricate structures and interactions. Dynamic community detection (DCD) is essential for understanding evolving complex networks, where vertices and edges continuously change over time. However, maintaining stability and continuity in dynamic networks remains a significant challenge, as communities experience birth, death, splitting, or merging. Existing techniques often struggle with these evolving dynamics, causing inconsistent tracking and reduced stability. To address these limitations, this thesis proposes an enhanced incremental dynamic community detection called Dynamic Community Detection based on the Bird Flock Effect (DCDBFE). This technique was inspired by the behavior of bird flocks and utilizes the principles of separation, alignment, and cohesion. The proposed technique incorporates a Resource Allocation similarity measure and third-level module attraction function to dynamically update community structures. This incremental approach effectively captures temporal changes and ensures continuous community structure tracking, unlike traditional static models. To ensure rigorous evaluation, synthetic networks were generated using an extended Lancichinetti-Fortunato-Radicchi benchmark, specifically designed in three scales, low, medium, and large with controlled parameters for mixing coefficient, edge density, and probability of vertex switching. These network datasets enabled systematic assessment of module stability across different network complexities. The experimental results were further validated using real-world network datasets to confirm the stability of the proposed technique. Stability was quantified based on inter-snapshot similarity, using metrics such as the Normalized Mutual Information and Adjusted Rand Index. Statistical validation was conducted using the Friedman test with a 95% confidence level to confirm the significance of performance improvements among competing techniques. The results demonstrated that DCDBFE achieved up to 15-25% higher stability compared to state-of-the-art DCD techniques. This research contributes to the field of dynamic community detection by providing a stable solution through an innovative methodology inspired by natural phenomena. Future work will extend DCDBFE to heterogeneous and weighted networks, and integrate deep learning-based adaptive similarity functions to further enhance temporal prediction and real-time community detection analysis.

PENGESANAN KOMUNITI DINAMIK SECARA TOKOKAN TERTINGKAT UNTUK MENAMBAHBAIK KESTABILAN STRUKTUR KOMUNITI DALAM ANALISIS RANGKAIAN

ABSTRAK

Analisis rangkaian memainkan peranan penting dalam mengenal pasti komuniti rangkaian kompleks merentasi pelbagai bidang, sekali gus mendedahkan struktur serta interaksi yang rumit. Pengesanan komuniti dinamik (DCD) amat penting untuk memahami evolusi struktur rangkaian kompleks, di mana bucu dan sisi sentiasa berubah dari semasa ke semasa. Walau bagaimanapun, mengekalkan kestabilan serta kesinambungan dalam rangkaian dinamik merupakan cabaran utama kerana komuniti mengalami pembentukan, pembubaran, pemecahan atau penggabungan. Teknik sedia ada sering menghadapi kesukaran dalam menangani dinamik yang berubah-ubah, menyebabkan penjejakan yang tidak konsisten dan kestabilan berkurangan. Bagi menangani kekangan tersebut, tesis ini mencadangkan satu teknik pengesanan komuniti dinamik secara tokokan tertingkat bernama Pengesanan Komuniti Dinamik berdasarkan Kesan Kawanan Burung (DCDBFE). Teknik ini diilhamkan daripada tingkah laku kawanan burung berdasarkan prinsip pemisahan, penjajaran dan kohesi. Teknik yang dicadangkan ini menggabungkan ukuran kesamaan Peruntukan Sumber dan fungsi tarikan modul peringkat ketiga bagi mengemas kini struktur komuniti secara dinamik. Pendekatan tokokan ini berupaya mewakili perubahan temporal dengan berkesan dan memastikan penjejakan struktur komuniti secara berterusan, berbeza dengan model tradisional yang statik. Bagi memastikan penilaian yang menyeluruh, rangkaian sintetik telah dijana menerusi penanda aras Lancichinetti-Fortunato-Radicchi yang diperluas dan dibina dalam tiga skala iaitu rendah, sederhana dan besar, dengan parameter terkawal meliputi pekali pencampuran, ketumpatan sisi dan kebarangkalian pertukaran bucu. Set data rangkaian tersebut membolehkan penilaian kestabilan modul yang sistematik mengikut tahap kerumitan rangkaian yang berbeza. Keputusan eksperimen turut disahkan melalui set data rangkaian dunia sebenar untuk membuktikan kestabilan teknik ini. Kestabilan diukur menggunakan kesamaan antara rentas masa seperti Maklumat Saling Ternomal dan Indeks Rand Terlaras. Pengesahan secara statistik telah dilakukan dengan kaedah ujian Friedman pada tahap keyakinan 95% bagi menunjukkan kepentingan peningkatan prestasi berbanding teknik lain. Hasil kajian mendapati bahawa DCDBFE mencapai peningkatan kestabilan sebanyak 15-25% lebih tinggi berbanding teknik DCD sedia ada. Penyelidikan ini menyumbang kepada bidang pengesanan komuniti dinamik dengan menyediakan penyelesaian yang stabil melalui satu metodologi inovatif yang diinspirasikan oleh fenomena semula jadi. Kajian masa hadapan akan memperluaskan penggunaan DCDBFE kepada rangkaian heterogen dan pemberat, serta mengintegrasikan fungsi kesamaan adaptif berasaskan pembelajaran mendalam bagi meningkatkan keupayaan ramalan temporal dan analisis evolusi komuniti secara masa nyata.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my supervisor, Prof. Madya Ts. Dr. Zuraida Abal Abas, and my co-supervisor, Dr. Norhazwani Mohd Yunos, for their invaluable guidance, patience, and support. May Allah SWT bless them for their kindness and dedication.

To my beloved husband, ‘Umar bin Ami, thank you for your unwavering love, support and belief in me. To my children, ‘Ali, Hamzah, Bilal, and Siti Khadijah, you are my greatest motivation. This thesis is dedicated to you, a reminder that with trust in Allah SWT and perseverance, anything can be reached.

I am deeply grateful to my late father, Hj Hairol Anuar—may Allah grant him Jannah—and my lovely mother, Halimah binti Hj Hasim, for their endless prayer and support. To my siblings, thank you for being my strength and comfort throughout this journey.

Special thanks to my classmates, cohort members, and friends for their encouragement and prayers. I also appreciate the support received from FTMK and Bahagian Cuti Belajar UTeM, which made this journey possible.

Thank you all for the contribution to this achievement and for being part of this journey.

TABLE OF CONTENTS

		PAGES
DECLARATION		
APPROVAL		
DEDICATION		
ABSTRACT		**i**
ABSTRAK		**ii**
ACKNOWLEDGEMENT		**iii**
TABLE OF CONTENTS		**iv**
LIST OF TABLES		**vii**
LIST OF FIGURES		**viii**
LIST OF ABBREVIATIONS		**xv**
LIST OF SYMBOLS		**xvii**
LIST OF APPENDICES		**xix**
LIST OF PUBLICATIONS		**xx**
LIST OF CONFERENCES		**xxi**
CHAPTER		
1.	**INTRODUCTION**	**1**
	1.1 Introduction	1
	1.2 Research Background	1
	1.3 Problem Statement	4
	1.4 Research Questions	6
	1.5 Research Objectives	7
	1.6 Scope of Research	7
	1.7 Significance of Study	8
	1.8 Thesis Outline	9
2.	**LITERATURE REVIEW**	**12**
	2.1 Introduction	12
	2.2 Overview of Community Detection	13
	2.2.1 Basic Concept and Terminology	15
	2.2.2 Network Representation	16
	2.2.3 Community Structure	20
	2.2.4 Community Detection	21
	2.3 Process of Community Detection	26
	2.3.1 Similarity Measure for Community Detection	28
	2.3.2 Limitations of Similarity Measure in Community Detection	33
	2.3.3 Vertex Attraction for Community Detection	36
	2.3.4 Module Attraction for Community Detection	37
	2.3.5 Limitation of Module Attraction in Community Detection	40
	2.4 Overview of Dynamic Community Detection	43
	2.4.1 Independent Community Detection	50
	2.4.2 Dependent Community Detection	50
	2.4.3 Incremental Dynamic Community Detection	53

2.4.4 Inspiration from Natural Phenomena 59
2.4.5 Limitations in Existing Dynamic Community Detection 63
2.5 Behavioural and Event Used in Network Structure 64
2.6 Performance Evaluation for Dynamic Community Detection 69
2.6.1 Stability Metric 70
2.6.2 Ground Truth Network 72
2.7 Performance Validation using Statistical Approach 74
2.7.1 Friedman Test 75
2.8 Research Gap 76
2.9 Summary 77

3. RESEARCH METHODOLOGY 79
3.1 Introduction 79
3.2 Research Framework 79
3.2.1 Investigation Stage 81
3.2.2 Implementation Stage 81
3.2.3 Network Datasets 91
3.3 Comparison Between Existing and Proposed Technique 99
3.4 Summary 101

4. PROPOSED DYNAMIC COMMUNITY DETECTION 103
4.1 Introduction 103
4.2 Basic Idea 103
4.3 Relevant Definitions 105
4.3.1 Resource Allocation for Similarity Measure 106
4.3.2 Vertex Attraction 108
4.3.3 Module Attraction 109
4.4 Inspired Bird Flock Effect Model for Community Detection 113
4.5 Dynamic Community Detection Inspired by Bird Flock Effect 117
4.6 DCDBFE Pseudo Code 124
4.7 Summary 129

5. RESULT AND DISCUSSION 130
5.1 Introduction 130
5.2 Result Phase 1: Preliminary Similarity Analysis 130
5.2.1 Performance of Comparison Similarity Measures 131
5.2.2 Summary of Results in Phase 1 138
5.3 Result Phase 2: Module Analysis 139
5.3.1 Performance of Comparison Level Connection 140
5.3.2 Summary of Results in Phase 2 143
5.4 Result Phase 4: Evaluation of Proposed Dynamic Community Detection 143
5.4.1 Performance of Comparison Techniques on Synthetic Networks 144
5.4.2 Performance of Comparison Techniques on Real-World Networks 169
5.4.3 Summary of Results in Phase 4 177
5.5 Statistical Approach for Validation 177
5.5.1 Validation of Results in Phase 1 178
5.5.2 Validation of Results in Phase 2 180

5.5.3 Validation of Results in Phase 4 on Synthetic Networks 181
5.5.4 Validation of Results in Phase 4 on Real-World Networks 183
5.6 Summary 185

6. CONCLUSION AND RECOMMENDATIONS FOR FUTURE RESEARCH 187
6.1 Introduction 187
6.2 Thesis Summary 187
6.3 Research Contributions 189
6.4 Practical Implications and Beneficiaries 191
6.5 Limitations and Recommendations for Future Research Work 192

REFERENCES 194
APPENDICES 204

LIST OF TABLES

TABLE	TITLE	PAGE
Table 2.1	Adjacency list	18
Table 2.2	Adjacency matrix	19
Table 2.3	Comparison of similarity measures and module attractions used in existing DCD	42
Table 2.4	Summary of evolution and approaches of dynamic community detection	47
Table 2.5	Comparison aspects between incremental and evolutionary approaches in dependent dynamic community detection	52
Table 2.6	List of several existing dependent dynamic community detection	54
Table 2.7	Summary of performance evaluation and behaviour in dynamic network from previous studies	67
Table 2.8	Summary of evaluation criteria settings for stability issue	69
Table 3.1	Definition of parameters of the extended LFR	94
Table 3.2	Evaluation criteria, setting and purpose of experiment in synthetic networks	95
Table 3.3	The real-world network features in this study	99
Table 5.1	Performance validation of RA using Friedman test on various merger-split network datasets in Phase 1	178
Table 5.2	Performance validation of third level connections using Friedman test on merger and spit network datasets in Phase 2	180
Table 5.3	Performance validation of DCDBFE using Friedman test on various synthetic network datasets in Phase 4	181
Table 5.4	Performance validation of DCDBFE using Friedman test on various real-world network datasets in Phase 4	184

LIST OF FIGURES

FIGURE	TITLE	PAGE
Figure 2.1	Literature review flowchart	13
Figure 2.2	An undirected and unweighted network	16
Figure 2.3	Example of community structure in a network	20
Figure 2.4	Infographic of Louvain (Traag, Waltman and van Eck, 2019)	24
Figure 2.5	Infographic of Leiden (Traag, Waltman and van Eck, 2019)	25
Figure 2.6	Process in community detection	27
Figure 2.7	Usage timeline of similarity indices used in DCD (2000-2025)	34
Figure 2.8	Dimensions of community structure in complex networks	43
Figure 2.9	Evolution and classification of dynamic community detection	46
Figure 3.1	Research framework of dynamic community detection in this thesis	80
Figure 3.2	Phase 1: Similarity analysis by six types of similarity measure	83
Figure 3.3	Phase 2: Module analysis based on the level of connectivity	85
Figure 3.4	Illustration for level connection of module attraction	86
Figure 3.5	Phase 3: Proposed technique inspired by bird flock effect	88
Figure 3.6	Phase 4: Evaluation and validation of the proposed technique	90
Figure 3.7	Comparison between existing and proposed technique	100
Figure 4.1	The formation process of bird flock effect model	104
Figure 4.2	A toy network of of two distinct modules, A and B	106
Figure 4.3	Flowchart of community detection inspired by bird flock effect model	115
Figure 4.4	Framework of DCDBFE	118
Figure 4.5	Demonstration of the process of incremental community detection based on bird flock effect from time steps T_1 until T_8	119

Figure 5.1 Comparison of similarity measures based on NMI metric with 5 events of merger and 5 events of split (m5_s5) 133

Figure 5.2 Comparison of similarity measures based on ARI metric with 5 events of merger and 5 events of split (m5_s5) 133

Figure 5.3 Comparison of similarity measures based on NMI metric with 20 events of merger and 20 events of split (m20_s20) 134

Figure 5.4 Comparison of similarity measures based on ARI metric with 20 events of merger and 20 events of split (m20_s20) 135

Figure 5.5 Comparison of similarity measures based on NMI metric with 40 events of merger and 40 events of split (m40_s40) 136

Figure 5.6 Comparison of similarity measures based on ARI metric with 40 events of merger and 40 events of split (m40_s40) 136

Figure 5.7 Execution time for preliminary similarity analysis 137

Figure 5.8 Comparison of different level connection based on NMI metric with 40 events of merger and 40 events of split (m40_s40) 141

Figure 5.9 Comparison of different level connection based on ARI metric with 40 events of merger and 40 events of split (m40_s40) 142

Figure 5.10 Comparison of techniques based on NMI metric with 5, 20 and 40 events of merger and 5, 20 and 40 events of split (m5_s5/ m20_s20/ m40_s40) 145

Figure 5.11 Comparison of techniques based on ARI metric with 5, 20 and 40 events of merger and 5, 20 and 40 events of split (m5_s5/ m20_s20/ m40_s40) 148

Figure 5.12 Comparison of techniques based on NMI metric with 5, 20 and 40 events of expansion and 5, 20 and 40 events of contraction (e5_c5/ e20_c20/ e40_c40) 151

Figure 5.13 Comparison of techniques based on ARI metric with 5, 20 and 40 events of expansion and 5, 20 and 40 events of contraction (e5_c5/ e20_c20/ e40_c40) 153

Figure 5.14 Comparison of techniques based on NMI metric with 2, 8 and 16 events of birth and 2 and 8 events of death (b2_d2/ b8_d8/ b16_d8)156

Figure 5.15 Comparison of techniques based on ARI metric with 2, 8 and 16 events of birth and 2, and 8 events of contraction (b2_d2/ b8_d8/ b16_d8) 158

Figure 5.16 Comparison of techniques based on NMI metric with mixing parameter, 160

Figure 5.17 Comparison of techniques based on ARI metric with mixing parameter, 161

Figure 5.18 Comparison of techniques based on NMI metric with average degree k = 5, 15, 25 (k5/ k15/ k25) 163

Figure 5.19 Comparison of techniques based on ARI metric with average degree k = 5, 15, 25 (k5/ k15/ k25) 164

Figure 5.20 Comparison of techniques based on NMI metric with probability 166

Figure 5.21 Comparison of techniques based on ARI metric with probability 169

Figure 5.22 Comparison of techniques based on NMI and ARI metric for workplace contact 170

Figure 5.23 Comparison of techniques based on NMI and ARI metric for primary school network 171

Figure 5.24 Comparison of techniques based on NMI and ARI metric for high school in 2011 network 173

Figure 5.25 Comparison of techniques based on NMI and ARI metric for high school in 2012 network 174

Figure 5.26 Comparison of techniques based on NMI and ARI metric for cumulative coauthor 175

Figure 5.27 Comparison of techniques based on NMI and ARI metric for non-cumulative coauthor 176

LIST OF ABBREVIATIONS

AA	-	Adamic Adar
ARI	-	Adjusted Random Index
CC	-	Cumulative coauthorship
CD	-	Community detection
CDBFE	-	Community detection based on bird flock effect
CDME	-	Community detection based on Matthew effect
CDFSE	-	Community detection based on fish school effect
CN	-	Common neighbors
DyPerm	-	Dynamic community Detection by maximizing Permanence
DCD	-	Dynamic community detection
DCDBFE	-	Dynamic community detection based on bird flock effect
DCDME	-	Dynamic community detection based on Matthew effect
DCDID	-	Dynamic Community Detection based on Information Dynamic
DYNMOGA	-	Dynamic Multi-Objective Genetic Algorithms
FacetNet	-	Framework for Analyzing Communities and EvoluTions in dynamic NETworks
HS2011	-	High school 2011
HS2012	-	High school 2012
IncNSA	-	Incremental Node Similarity Algorithm
LFR	-	Lancichinetti-Fortunato-Radicchi
LHN	-	Leicht-Holme-Newman
NCC	-	Non-cumulative coauthorship

NMI	-	Normalize Mutual Index
PS	-	Primary school
QCA	-	Quick community adaptation
RA	-	Resource Allocation
$UTeM$	-	Universiti Teknikal Malaysia Melaka
WC	-	Workplace contact

LIST OF SYMBOLS

$G(V, E)$	-	A set of network or graph with pair of vertex of V and edge of E
V	-	A set of vertex V
E		A set of an edge E
$G(v_i, e_i)$	-	A network or graph with vertex v of i and edge e of i
v_i	-	A vertex of i
e_i	-	An edge of i
n	-	Number of vertices in G or each time step
e	-	Number of edges in G
$D(u)$	-	Degree of vertex u
A	-	Adjacency matrix
A_{ij}	-	Adjacency matrix with edge between vertices v_i and v_j,
M	-	Module
DCD	-	Dynamic community detection
m_{sub_u}	-	Submodule of vertex u
$\Gamma(v_i)$	-	Set of neighborhood vertex of v_i
RA	-	Resource allocation similarity index
$VA_{v \to u}$	-	Vertex attraction from vertices u and v
$MA_{v \to A}$	-	Module attractiveness from vertex v to module A
ts	-	Time step

ts-1	-	Previous time step
S	-	Number of time steps in the dynamic network
μ	-	Mixing parameter
k	-	Average degree of the dynamic network
p	-	Probability of vertex switching between two adjacent time steps
e	-	Number of community expands in every time step
c	-	Number of community contracts in every time step
b	-	Number of community births in every time step
d	-	Number of community deaths in every time step
m	-	Number of community mergers in every time step
s	-	Number of community splits in every time step
V_{AV}	-	Addition vertices event
V_{DV}	-	Deletion vertices event
E_{AE}	-	Addition edges event
E_{DE}	-	Deletion edges event
$N(v)$	-	Neigbour of vertex v

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
Appendix A	Coding for DCDBFE	204
Appendix B	Coding for PHASE 1 Similarity Analysis	207
Appendix C	Coding for PHASE 2 Module Analysis	211
Appendix D	Coding for PHASE 3 Bird Flock Effect	212
Appendix E	Coding for PHASE 4 Evaluate DCDBFE	214
Appendix F	Coding for Validation using Statistical Approach	222

LIST OF PUBLICATIONS

The followings are the list of publications related to the work on this thesis:

S. H. H. Anuar, Z. A. Abas, N. M. Yunos, M. F. Mukhtar, T. Setiadi, and A. S Shibghatullah, 2024. Identifying Communities with Modularity Metric using Louvain and Leiden Algorithms. *Pertanika Journal of Science and Technology, UPM*. Vol. 32 (3), pp. 532–537, 2024. (ISI indexed, Q3, IF = 0.6) (SCOPUS indexed). https://doi.org/10.47836/pjst.32.3.16

S. H. H. Anuar, Z. A. Abas, N. M. Yunos, M. F. Mukhtar, and N. H. Miswan, 2024. Recent trends on practical applications in community detection: A Thematic Review. *International Journal of Academic Research in Business and Social Sciences*. Vol. 14 (10), pp. 953-996, 2024. (ERA indexed) http://dx.doi.org/10.6007/IJARBSS/v14-i10/23160

S. H. H. Anuar, Z. A. Abas, I. Waini, M. F. Mukhtar, Z. Sun, E. A. Winanto, and N. M. Yunos. 2025. Bird Flock Effect-Based Dynamic Community Detection: Unravelling Network Patterns Over Time. *Alexandria Engineering Journal, Elsevier*. (ISI indexed, Q1, IF = 6.8) (SCOPUS indexed) https://doi.org/10.1016/j.aej.2024.10.097

LIST OF CONFERENCES

The followings are the list of conferences related to the work on this thesis:

S. H. H. Anuar, Z. A. Abas, N. M. Yunos, N. H. M. Zaki, N. A. Hashim, M. F. Mukhtar, S. A. Asmai, Z. Z. Abidin, and A. F. Nizam, 2021. Comparison between Louvain and Leiden Algorithm for Network Structure: A Review, in *Journal of Physics: Conference Series,* IOP. Publishing. ICAC2021. 4-5 February 2021. Vol. 2129 (1)*,* pp. 012028. (SCOPUS indexed). https://doi.org/10.1088/1742-6596/2129/1/012028

S. H. H. Anuar, Z. A. Abas, N. M. Yunos, N. H. M. Zaki, M. F. Mukhtar. 2022. TRENDS ON TRACKING DYNAMIC COMMUNITY DETECTION: 2017-2021. Seminar on Information Retrieval and Knowledgement 2022 (SIRKM 22 UKM). 2-3 March 2022. https://ftsm.ukm.my/sirkm2022/sirkm2022/images/E-Proceeding%20SIRKM22.pdf

CHAPTER 1

INTRODUCTION

1.1 Introduction

Chapter 1 introduces the foundation of this research, outlining its significance, objectives, and scope. It begins by discussing the growing importance of network analysis in complex systems. It emphasizes extracting meaningful insights through community analytics and highlights the associated challenges in the research background section. The chapter then presents the problem statement, focusing on the stability issues in detecting dynamic communities within networks. The study's objectives are well-defined, aiming to propose an enhanced technique for dynamic community detection (DCD) inspired by natural phenomena. Additionally, the chapter outlines the methodology and approach employed throughout the research, providing a roadmap for the subsequent chapters.

1.2 Research Background

A wide variety of systems in nature, society, and technology can be modeled as networks (Holme and Saramäki, 2019). Over the past two decades, studies on networks have expanded significantly, with network sizes growing to include millions or even billions of vertices. This is because networks are essential part of everyday life, providing a convenient way to represent data across various domains, such as market systems, transportation systems, biological systems, and others (Christopoulos et al., 2023). The growth of networks is fueled by technological advancements, widespread digitalization, and the increasing

availability of large-scale datasets from diverse fields (Khawaja et al., 2024). Examples of such networks include Twitter networks, smart city networks, protein interaction networks, collaboration networks, and the Enron email network. This has significant implications for the study of network structures, known as network analysis.

According to SCOPUS data from June 2024, the number of research studies on network analysis has grown by over 150,000 publications since 1994. However, as networks continue to grow in size, visual representation and analysis become increasingly impractical, prompting researchers to focus on mathematical descriptions of network structures. Despite this shift, network analysis techniques are needed to handle large-scale networks (Khawaja et al., 2024).

Network analysis, also referred to as graph analytics, is a powerful tool for measuring relationships and interactions within complex systems. The terms "network analysis" and "graph analytics" can be used interchangeably. Graph analytics is founded on linear algebra, which is essential for students in science, engineering, and mathematics due to its wide range of real-life applications (Zarayeneh and Kalyanaraman, 2021; Hairol Anuar et al., 2024; Nakos, 2024). There are four main types of graph analytics: Path analytics, which determines the shortest distance between vertices; Connectivity analytics, which identifies weaknesses within networks; Centrality analytics, which determines the most influential vertices; and Community analytics, which identifies groups of closely connected vertices.

Community analytics is effective in revealing the structure of communities within a network, where internal connections are denser than external ones. A critical task in community analytics is community detection, which identifies groups or modules with more frequent interactions among members than with the rest of the network. This process offers

insights into network structure, helping researchers understand relationships, analyze influence patterns, recommend appropriate products, and organize complex systems (Karatas and Sahin, 2018). Most literature focus on community detection in static networks, where the structure remains unchanged. However, real-world networks are dynamic, evolving over time in terms of vertices, edges, and attributes. As a result, many existing community detection techniques are inadequate for dynamic networks, driving increased interest in DCD (Choudhury, 2024).

DCD has emerged as a significant area of research due to its ability to analyze the temporal evolution of social phenomena, as highlighted by Karatas & Sahin (2022). This research is relevant in applications like social media and online networks, where rapid changes require techniques capable of adapting to dynamic network structures and processing data in real time. Over the past decade, various DCD techniques have been employed to identify high-quality community structures based on different evaluation metrics. The techniques commonly used in DCD include modularity, label propagation, spectral clustering, non-negative matrix factorization (NMF), multi-objective approaches, and others (Aynaud et al., 2013; Dakiche et al., 2019; Rossetti and Cazabet, 2019; Christopoulos and Tsichlas, 2022). While these techniques offer diverse approaches, each has its weaknesses depending on the complexity and type of data processed.

Identifying and utilizing influential vertices is key to improving DCD accuracy and stability, as these vertices often drive community formation and evolution (Z. Sun et al., 2022). Stable DCD requires refining similarity measures and using connectivity patterns as module attraction for better evaluation (Dai et al., 2019). Both are central to assessing the quality of detected community structures, but their optimization remains an ongoing research

challenge. Errors in similarity measurement can lead to flawed DCD and poor decision-making, particularly in real-time applications such as social networks. Therefore, maintaining consistency in similarity measures and module attraction as communities evolve ensures that the communities remain cohesive.

1.3 Problem Statement

The increasing volume and complexity of data in modern networks pose significant challenges for the development of stable and adaptive dynamic community detection (DCD) techniques. As networks evolve through the continuous addition and deletion of vertices and edges, maintaining stable and consistent community structures becomes difficult (Bouhatem et al., 2021; Zarayeneh and Kalyanaraman, 2021). Although recent approaches have improved adaptability, most existing techniques still suffer from instability, where detected communities fluctuate significantly between time steps despite minor structural changes (Zhuang, Chang and Li, 2019). This instability makes it difficult to interpret the true evolution of communities across time.

Existing DCD techniques can be broadly categorized into independent and dependent techniques. Independent community detection performs community detection separately on each snapshot, treating every time step as an isolated network. This results in low temporal consistency because past structural information is not considered. In contrast, dependent community detection maintains continuity between time steps by incorporating temporal relationships. These dependent techniques are mainly divided into evolutionary and incremental approaches. Evolutionary approaches attempt to balance historical and current partitions to improve temporal smoothness. However, they often sacrifice modularity or

intra-step quality in exchange for stability. Incremental approaches update community structures dynamically as the network evolves, achieving higher stability but still facing challenges in efficiently balancing modularity (quality at each time step) and stability (consistency across time), especially in highly dynamic environments (F. Liu et al., 2020).

Instability in DCD has been quantified using inter-step metrics such as Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI), which measure similarity between consecutive community partitions (Rossetti et al., 2018). Lower values of NMI or ARI indicate greater instability, suggesting that the detected community structures vary significantly between time steps (Cazabet et al., 2021). While these metrics provide a numerical indication of instability, most studies lack standardized statistical validation, making it difficult to determine whether the instability is inherent to the method or caused by network dynamics.

Another critical limitation in current community detection is reliance on basic representations such as similarity measures and connectivity patterns (Su et al., 2020; Sun, Sheng, et al., 2020). Most of existing techniques that adopt similarity measures such as Jaccard and Salton are crucial for identifying intra-community similarities. The measures are useful for identifying direct neighbour overlaps but fail to capture higher-order or temporal relationships among vertices. As a result, vertices with weak or indirect connections are still less optimum which may be misclassified during community transitions, producing unstable similarity measures and inconsistent community tracking (Ghawi and Pfeffer, 2022; Choudhury, 2024).

Recent studies have explored natural phenomena such as swarm intelligence and behavioral models to improve the adaptability of DCD techniques. These phenomena, if

incorporated, could improve the adaptability and stability of models, but current techniques lack effective frameworks to integrate them (Rahimi et al., 2018). Incorporating natural dynamics, particularly module attraction which reflects the tendency of vertices or modules to move closer over time. Nonetheless, the current module attractions typically consider only limited levels of connectivity, reducing their impact on overall stability (Z. Sun et al., 2022).

Therefore, this thesis aimed to improve the stability issues in dynamic community detection by integrating similarity-based techniques with module attraction process and leveraging natural phenomena within a statistical evaluation approach.

1.4 Research Questions

Three research questions were investigated to propose a dynamic community detection technique in network structure.

i) How to determine an appropriate similarity measure and enhanced module attraction for capturing both local and vertices relationship in dynamic networks?

ii) How to propose an enhanced dynamic community detection technique based on incremental approach inspired by bird flock effect for ensuring temporal stability?

iii) How to evaluate the proposed technique in objectives 1 & 2 by comparison with existing well-known techniques using non-parametric statistical approach such as Friedman test for quantitative assessment?

1.5 Research Objectives

The main aim of this research was to propose an enhanced dynamic community detection technique for network structures for improving stability. Specifically, the research objectives were as follows:

i) To determine an appropriate similarity measure and enhanced module attraction for capturing both local and vertices relationship in dynamic networks.

ii) To propose an enhanced dynamic community detection technique based on incremental approach inspired by bird flock effect for ensuring temporal stability.

iii) To evaluate the proposed technique in objectives 1 & 2 by comparison with existing well-known techniques using non-parametric statistical approach such as Friedman test for quantitative assessment.

1.6 Scope of Research

The scope of this study was focused on the development and evaluation of a technique, Dynamic Community Detection based on Bird Flock Effect (DCDBFE), for addressing stability challenges in evolving networks. The research focused on unweighted and undirected dynamic networks, including the addition or deletion of vertices and edges, leading to community events like birth, death, merging, and splitting. It adopted a dependent incremental approach, integrating an enhanced similarity attraction framework that combines optimized similarity measures with a multi-level module attraction to maintain stable and capture both local and vertices relationships. The proposed technique was inspired by principles of bird flock behavior, specifically separation, alignment, and cohesion. The

research did not consider heterogeneous, weighted or attribute-based networks. The proposed technique was tested on synthetic network datasets with varying scales and real-world networks, including social and collaboration networks. In total, 17 synthetic networks and 6 real-world networks were used in the experiments. Statistical approaches, including the Friedman test was applied to validate the results, demonstrating the technique's stability.

1.7 Significance of Study

The significance of this research can be classified into three categories: theoretical, methodological, and empirical. Theoretically, it contributes to the body of knowledge regarding dynamic community detection. By introducing an approach inspired by natural phenomena such as the behavior of bird flocks, the study expands current knowledge. It enhances the understanding of dynamic community detection, specifically on how communities form, evolve, and dissolve in ever-changing networks. It addresses key challenges, such as instability in dynamic detection techniques and inaccuracies caused by inappropriate measurements between vertices, or module attraction. Therefore, by providing a theoretical foundation, this research aimed to achieve more stable and cohesive community structures in undirected and unweighted networks.

Methodologically, the research offers an innovative framework for detecting dynamic communities, inspired by the natural behavior of bird flocks. It incorporated three fundamental behaviors observed in flocking birds, specifically separation (staying apart), alignment (moving together), and cohesion (sticking close) to create a stable and adaptable approach. This framework refines similarity measures and module attraction, addressing limitations in existing techniques and enhancing the stability of community detection over

time. The methodology based on natural phenomena makes it uniquely suited to tackle the complexities of dynamic and noisy networks.

Empirically, the study validated the proposed technique through rigorous testing on both synthetic and real-world networks. These include social and collaboration networks, where fast-changing environments require accurate and stable dynamic community detection. By comparing the proposed technique with several well-known existing techniques, the research highlights its strengths and practical applicability. The results demonstrate how this technique can effectively analyze interaction patterns and network dynamics, particularly in real-time applications like social media. Through its theoretical insights, methodological innovations, and empirical validation, this research significantly advances the field of dynamic community detection.

1.8 Thesis Outline

This thesis has six (6) chapters, which are based on the research objectives outlined in Section 1.5. Here is a summary of the content that each chapter includes:

- Chapter 1. Introduction. This chapter presents the research background, problem statement, research questions, objectives, scope, and significance of the study. It serves as an essential introduction, capturing the reader's attention and establishing the context for the research.
- Chapter 2. Literature Review. This chapter provides an extensive overview of the literature on network analysis, starting with community detection, its basic process, strengths, and limitations. It then delves into dynamic community detection, which addresses evolving networks by examining

independent and dependent detection techniques. The chapter highlights the influence of behaviors and events on network structures, drawing inspiration from natural phenomena. It also explains several stability measures and statistical approaches used for performance evaluation and validation in dynamic community detection. Finally, the research gap is emphasized in depth.

- Chapter 3. Research Methodology. This chapter provides a detailed explanation of the research methodology, outlining the step-by-step workflow and careful planning involved in the study. It starts with the research framework, which serves as the methodology to guide the entire research pipeline. The chapter then explains the network datasets and compares the existing and proposed techniques.
- Chapter 4. Proposed Dynamic Community Detection. This chapter outlines the basic idea, relevant definitions, flowchart, and the proposed dynamic community detection technique inspired by the bird flock effect in network structures. The core idea behind this technique is to apply natural phenomena that replicate the formation process of bird flocks, enabling the analysis of their temporal evolution. The chapter then discusses the pseudocode used throughout.
- Chapter 5. Result and Discussion. This chapter presents the analyzed and interpreted findings, offering valuable insights into the effectiveness of the proposed technique. The technique was evaluated by comparing it with well-

known existing techniques using non-parametric statistical approaches, such as the Friedman test, on both synthetic and real-world networks.

- Chapter 6. Conclusion and Recommendations for Future Research. The final chapter highlights the key findings of the research, demonstrating how the results align with the initial objectives and contribute to the understanding of dynamic community detection. It also provides recommendations for future research, emphasizing areas where further exploration could enhance the methodology and its application in dynamic network analysis.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Chapter 2 provides a comprehensive literature review for this study. The chapter begins by introducing the basic concepts, terminology, and techniques used to represent and analyze networks, emphasizing the significance of community structures. It explores the basic processes in community detection, including similarity measure as vertex attraction-based techniques, while discussing their strengths and limitations. Additionally, the chapter delves into dynamic community detection (DCD), which addresses evolving networks by examining independent and dependent detection techniques. The chapter highlights the influence of behaviors and events on network structures, drawing inspiration from natural phenomena. This chapter also explains several stability metrics and statistical approaches used for performance evaluation in DCD. Finally, the research gaps are emphasized in depth. Figure 2.1 presents a comprehensive literature review flowchart for this study.

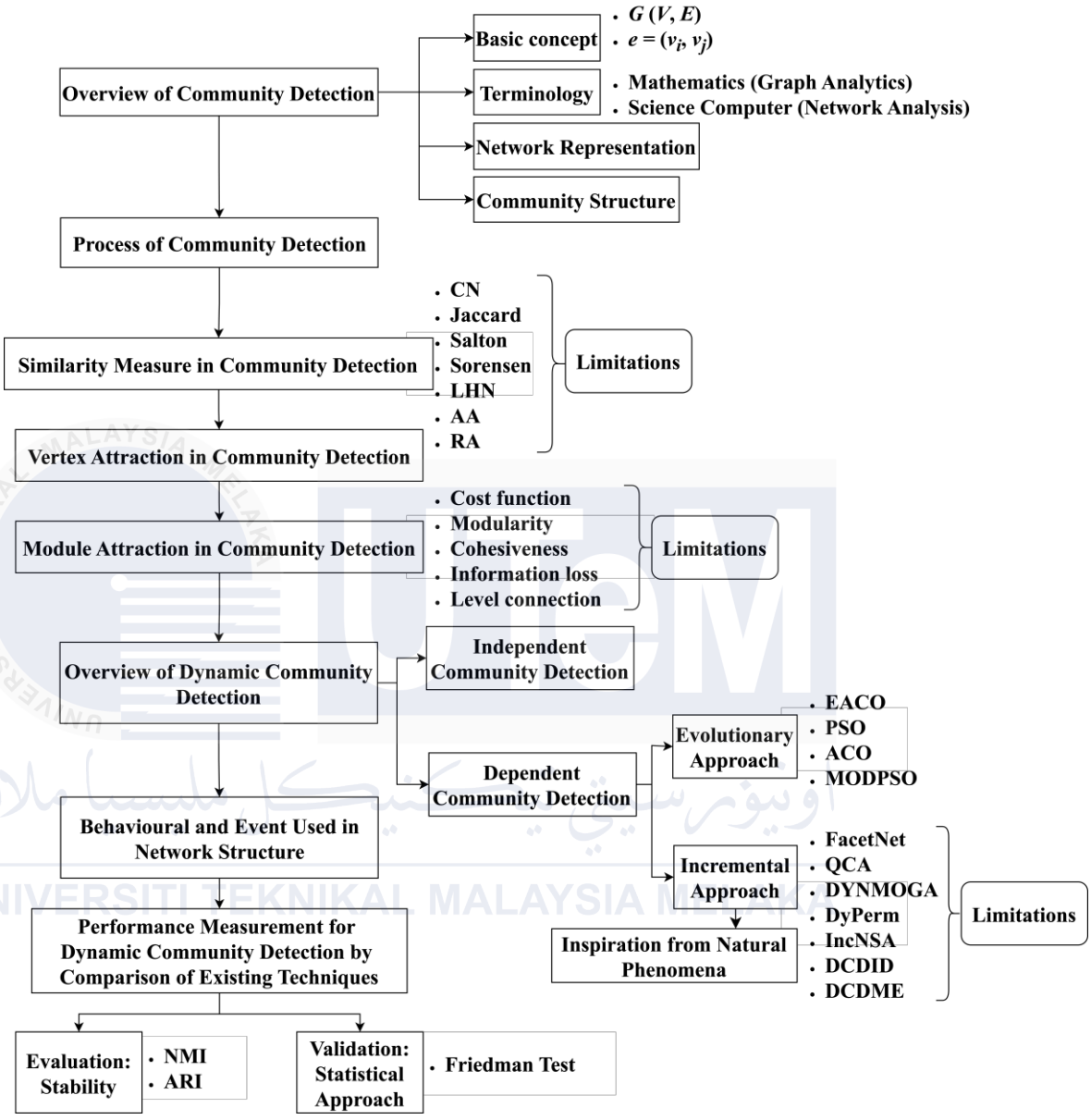


Figure 2.1 Literature review flowchart

2.2 Overview of Community Detection

Network analysis, also known as graph analytics, is a rapidly growing field that focuses on studying the relationships and interactions within complex systems (Holme and Saramäki, 2011). The mathematical foundation of network analysis lies in graph theory and linear algebra. It provides the necessary tools to model, analyze, and interpret the intricate

patterns within these networks. As networks continue to grow in size and complexity, the need for advanced analytical techniques has become increasingly important (Joshi and Patalia, 2020).

One of the key areas of focus within network analysis is community detection, which involves identifying groups of vertices within a network that are more densely connected to each other than to the rest of the network. These communities, also known as modules, often represent meaningful substructures within the network, such as social circles in a social network, functional groups in a biological network, or thematic clusters in an information network. Detecting these communities is crucial for understanding the organization and dynamics of complex networks, as it reveals the underlying structure that governs how vertices interact with one another (Nooribakhsh et al., 2024).

Over the years, numerous community detection techniques have been developed, and each of them has its own strengths and limitations. These techniques range from traditional methods based on modularity maximization and spectral clustering to more recent approaches that incorporate dynamic and hierarchical aspects of networks. As networks evolve over time, communities also change, leading to the emergence of dynamic community detection, a subfield dedicated to tracking and analyzing changes in community structure. This dynamic perspective is essential for applications where networks are not static, such as in social media, where relationships and interactions constantly shift. As the field of community detection continues to advance, it plays a pivotal role in unraveling the complexities of modern networks, enabling more informed decisions and deeper understanding of real-world systems (Bouhatem et al., 2021; Alotaibi and Rhouma, 2022; Christopoulos and Tsichlas, 2022).

2.2.1 Basic Concept and Terminology

To understand community detection, it's important to first grasp some basic concepts and terminologies used in network analysis. A network, also known as a graph in mathematical terms, consists of vertices (or nodes or points) and edges (or links or arcs) that connect pairs of vertices (Naoki Masuda and Renaud Lambiotte, 2021). These vertices can represent various entities such as individuals in a social network, proteins in a biological network, or webpages on the internet, while the edges signify the relationships or interactions between them. In the context of community detection, the goal is to identify subsets of vertices, known as communities, modules, or clusters, where the connections within each subset are denser than those between different subsets. These communities often correspond to functional groups, social circles, or other meaningful substructures within the network. Therefore, community detection is a critical tool for uncovering hidden patterns and understanding the underlying organization of complex networks (Rossetti, 2015).

As mentioned before, in mathematical context, network analysis also refers to graph theory involving specific terminology that defines the structure and elements of a network or graph. A graph G is represented as a pair $G(V, E)$, where V is the set of vertices and E is the set of edges connecting pairs of vertices. An edge e in E can be represented as a pair $e = (v_i, v_j)$, indicating that vertex v_i is connected to vertex v_j (Gulbahce and Lehmann, 2008; Papadopoulos et al., 2012; Shao, Han and Yang, 2014; Nooribakhsh et al., 2024).

The degree of a vertex v is the number of edges connected to it, which is mathematically expressed as $D(v)$. An undirected network only shows the interaction or relationship between vertices. Meanwhile, a directed network shows that each edge has a

direction, leading to the concepts of in-degree and out-degree, which represent the number of incoming and outgoing edges for a vertex, respectively. This thesis covers undirected and unweighted networks.

In community detection, a community or module is a subset of vertices $M \subseteq V$, where the number of edges within the subset is higher than the number of edges between the subset and the rest of the graph. Mathematically, the quality is often quantified using metrics like modularity, which measures the density of edges inside communities compared to what would be expected in a random graph. The adjacency matrix A of a graph is a square matrix where $A_{ij} = 1$ if there is an edge between vertices v_i and v_j, and $A_{ij} = 0$ otherwise. This matrix plays a crucial role in various techniques for detecting communities, particularly in methods involving spectral clustering or eigenvector analysis (Karatas and Sahin, 2022).

2.2.2 Network Representation

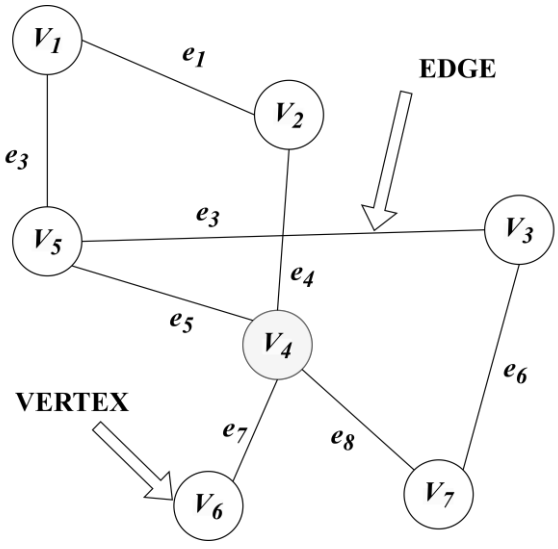


Figure 2.2 An undirected and unweighted network

As illustrated in Figure 2.2, an undirected and unweighted network consists of vertices, interconnected by edges, which signify interactions or similarities between them. The definition of vertices and edges in a network can vary depending on the specific type of network being considered (Holme and Saramäki, 2019). In this network, the vertices V_1 until V_7 represent individual entities, and the edges between them indicate connections or relationships. Since the network is undirected, the edges do not have any direction and the relationship between any two connected vertices is mutual. Additionally, being unweighted implies that all edges carry the same importance or strength; and there is no differentiation in the connection strength between any pair of vertices.

Figure 2.2 can also be represented in a different form known as an adjacency list, which is a common way to describe the connections between vertices in a network. An adjacency list provides a compact and efficient way to store graph data, especially for sparse graphs, where not every vertex is connected to every other vertex. In this representation, each vertex is listed alongside its adjacent vertices (those that are directly connected by an edge).

Table 2.1 illustrates the adjacency list corresponding to the network in Figure 2.2. In this table, each row represents a vertex, and the adjacent vertices listed beside it are those that share a direct edge with the vertex in question.

Table 2.1 Adjacency list

Vertex	**Adjacency Vertices**	**Edge list (Connection)**	**Degree of vertex, $D(v)$**
V_1	V_2, V_5	$(V_1,V_2), (V_1,V_5)$	2
V_2	V_1, V_4	$(V_2,V_1), (V_2,V_4)$	2
V_3	V_5,V_7	$(V_3,V_5), (V_3,V_7)$	2
V_4	V_2, V_5, V_6, V_7	$(V_4,V_2), (V_4,V_5),(V_4,V_6), (V_4,V_7)$	4
V_5	V_1, V_3, V_4	$(V_5,V_1), (V_5,V_3), (V_5,V_4)$	3
V_6	V_4	(V_6,V_4)	1
V_7	V_3, V_4	$(V_7,V_3), (V_7,V_4)$	2

For example, vertex V_1 is connected to V_2 and V_5, so the adjacency list entry for V_1 shows V_2 and V_5 as its adjacent vertices. Similarly, V_4 is connected to V_2, V_5, V_6, and V_7, so all these vertices are listed next to V_4. This representation is particularly useful for methods that traverse the network, such as those used in community detection, as it allows for efficient access to each vertex's neighbors.

The $D(v)$ column shows the degree of each vertex, which is simply the count of adjacent vertices listed in the previous column. For example, vertex V_4 has four connections to V_2, V_5, V_6, and V_7, so its degree $D(v)$ is 4. This indicates that V_4 plays a central role in the network because it is connected to more vertices than any other vertex. On the other hand, vertices like V_1, V_2, and V_3 have a degree of 1, indicating they each connect to only one other vertex in the network.

The adjacency matrix (Adj) is another way to represent a network or graph. It is a square matrix used to indicate whether or not the pairs of vertices are adjacent (connected) in the graph. Each element of the matrix Adj_{ij} is 1 if there is an edge between vertex v_i and vertex v_j, and 0 if there is no edge. For the network shown in Figure 2.2, the adjacency matrix Adj is as follows in Table 2.2.

Table 2.2 Adjacency matrix

Adj	V_1	V_2	V_3	V_4	V_5	V_6	V_7	$D(v)$
V_1	0	1	0	0	1	0	0	2
V_2	1	0	0	1	0	0	0	2
V_3	0	0	0	0	1	0	1	2
V_4	0	1	0	0	1	1	1	4
V_5	1	0	1	1	0	0	0	3
V_6	0	0	0	1	0	0	0	1
V_7	0	0	1	1	0	0	0	2
$D(v)$	2	2	2	4	3	1	2	

For instance, in the Adj derived from Figure 2.2, the entry for vertex V_4 shows connections with vertices V_2, V_5, V_6, V_7, which is reflected by the presence of 1s in the respective positions in the matrix. The matrix is symmetric, which reflects the undirected nature of the network, if vertex V_4 is connected to V_2, then V_2 is also connected to V_4.

2.2.3 Community Structure

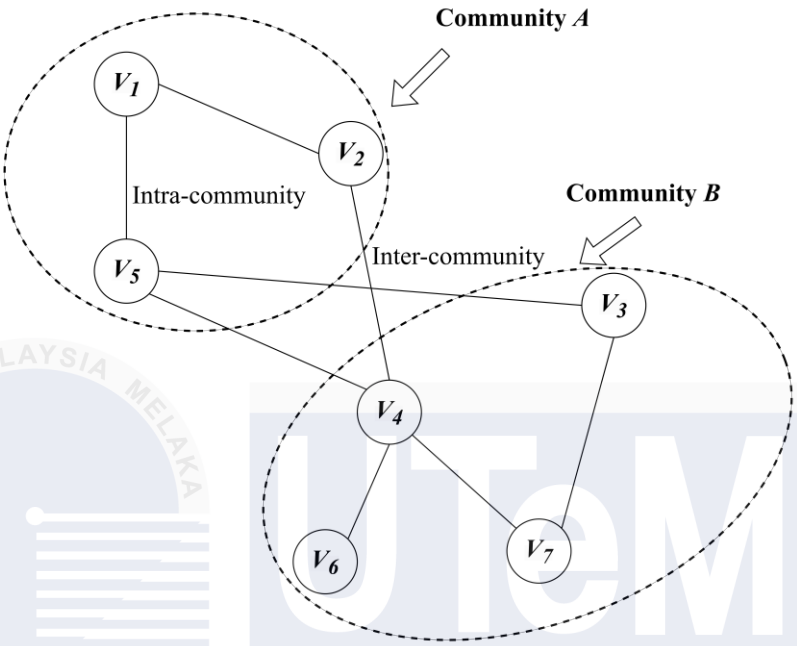


Figure 2.3 Example of community structure in a network

Figure 2.3 illustrates an example of community structure within a static network, which builds upon the basic concepts. In this network, the vertices are grouped into two distinct communities, labeled as Community A and Community B. The edge between two communities is called an inter-community edge. In the context of network analysis, a community is a subset of vertices that are more densely connected to each other than to the rest of the network. The community detection of vertices that move into communities reflects the idea that, within a community, the vertices share stronger or more frequent interactions, forming a cohesive substructure.

In this particular example, Community A consists of vertices V_1, V_2 , and V_5, which are more closely connected to each other than to vertices outside their module. Similarly,

Community B includes vertices V_3, V_4 , V_6, and V_7, which exhibit a denser pattern of connections among themselves. The edges connecting vertices within the same community represent stronger intra-community ties, whereas the connections between different communities are fewer and weaker, highlighting the modular nature of the network. This figure demonstrates the concept of community detection, which is a fundamental task in network analysis aimed at uncovering such modules within larger and more complex networks. Identifying these communities is crucial for understanding the underlying structure and function of networks, whether in social, biological, or technological systems.

2.2.4 Community Detection

In the field of network analysis, there are several fundamental tasks for understanding and interpreting the structure and behavior of complex networks. One of the most important tasks is community detection. This process involves identifying groups or communities of vertices within a network that are more densely connected to each other than to vertices outside their community. By uncovering these communities, researchers can gain insights into the modular structure of the network, which often reflects meaningful real-world groupings, such as social circles, functional modules in biological systems, or topic-based clusters in information networks. Understanding these communities is crucial for analyzing how different parts of the network interact and function together.

Community detection can be broadly divided into non-overlapping and overlapping approaches. Non-overlapping community detection assigns each vertex to exactly one community, forming a hard partition of the network. In contrast, overlapping community detection allows a vertex to belong to multiple communities at the same time, which reflects

many real-world scenarios. Most early techniques focus on non-overlapping communities, but recent studies show that overlapping structures provide a more realistic representation of complex networks.

Researchers have proposed various techniques and approaches, such as genetic algorithms, modularity, and nature-inspired methods, to identify communities within network structures. These community detection techniques are often categorized based on the approaches they use. For example, label propagation algorithm (LPA), Infomap, Louvain, Leiden, community detection based on the Matthew effect (CDME), and community detection based on the fish school effect (CDFSE), among others.

Raghavan, Albert, and Kumara (2007) present the label propagation algorithm (LPA), a fast and simple community detection approach. The LPA propagates labels adopted by network vertices until they converge. Labels indicate vertices' communities. Initial labels are assigned to each network vertex. Vertices will take the label most often used by their neighbors, with ties being broken randomly. Densely linked vertices naturally label and establish communities quickly. Labels are updated either synchronously or asynchronously. The propagation process continues until vertex labels no longer change. The LPA takes O(m) time for each propagation iteration on a network with m edges. After five repetitions, the labels converge significantly (Raghavan et al., 2007).

LPA propagation is guided only by the network structure. Apart from being a parameter-free detection approach, it does not require knowledge of the number or size of the communities. Additionally, its near-linear processing time makes it a useful technique for large networks. While LPA is simple and fast, it has certain downsides. When ties occur between multiple labels during the propagation process, they are broken randomly, which

makes the LPA non-deterministic and prevents it from yielding a unique detection result. Another drawback is its tendency to yield trivial detections (losing meaningful structure). The result of each iteration is not always stable.

Rosvall and Bergstrom (2008) proposed Infomap, a flow-based community discovery system that optimizes the map equation (Rosvall and Bergstrom, 2008). This approach utilizes the information theory concept such as compression. Infomap compresses and extracts important data patterns for naming the vertices. The idea is from Huffman (1952I in which a random walk's description length can be coded. Infomap finds network partitions that minimize random walk description length. The map equation calculates the average description length of a random walk step. However, infomap may suffer from a resolution limit where it fails to detect small communities within large networks and its performance heavily relies on the network's flow structure, which may not always be clear or well-defined. The map equation, which Infomap optimizes, can be complex and may not always be intuitive for interpreting the community structures.

Blondel et al. (2008) proposed a technique named Louvain which is the most popular in the greedy methodology approach since it surpassed most models in modularity scores and computational time (Blondel et al., 2008). Louvain has two stages. The first stage is the local moving of vertices for modularity optimization, while the second stage is the community merging or network aggregation process. The static network is required for Louvain to give an efficient output. In a large network, this approach, which belongs to the hierarchical clustering category, may quickly create communities with a high degree of modularity. As shown in the past study, this approach can find communities in 118 million vertices in 152 minutes. Figure 2.4 shows the infographic of Louvain.

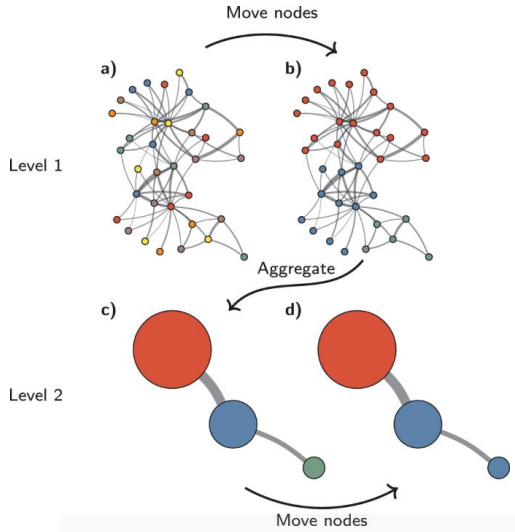


Figure 2.4 Infographic of Louvain (Traag, Waltman and van Eck, 2019)

A lot of community detections are based on the concept of modularity. However, Fortunato had proven that modularity cannot accurately evaluate small communities due to its resolution limit, which is biased against small communities (Blondel et al., 2008). In 2019, Vincent Traag et al. proposed the Leiden which is an enhancement of the Louvain (Traag, Waltman and van Eck, 2019). Despite the fact that it is more complicated than its counterpart, this technique is able to derive a faster and more precise computation time. As opposed to Louvain's, the Leiden comprises three phases, with the modularity optimization process being the first one, followed by the refinement of partition, and the community aggregation process in the last step as shown in Figure 2.5. In addition, this community detection works well on large-scale, medium, and small networks.

The model measures community structure locally and the resolution limit effects are not present in the technique. However, Leiden is a fast and excel algorithm yet prone to suffer from convergence to local optimum solutions.

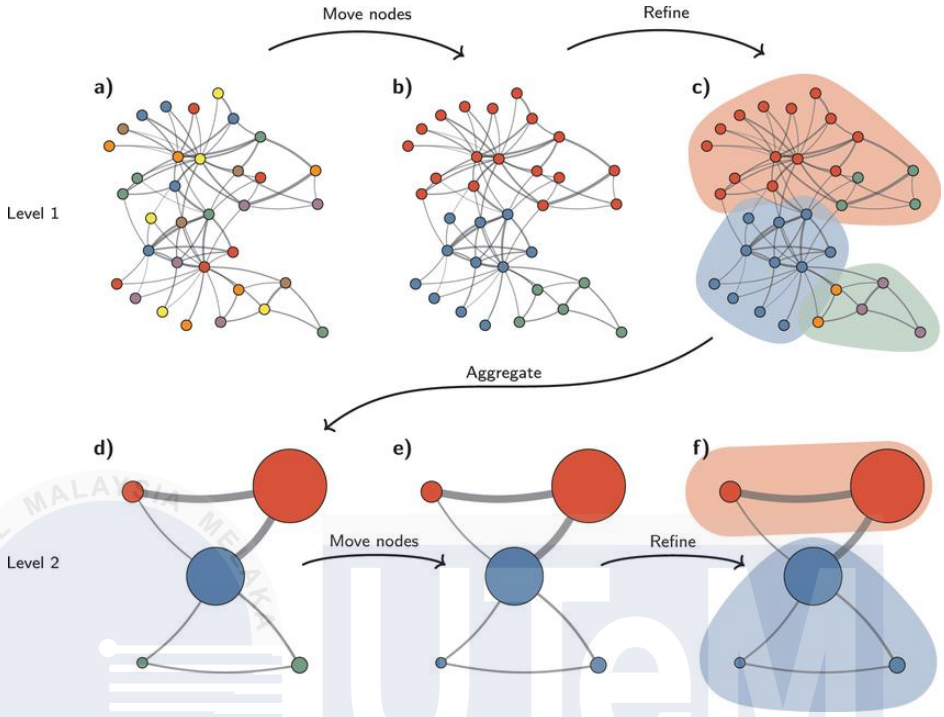


Figure 2.5 Infographic of Leiden (Traag, Waltman and van Eck, 2019)

Sun et al. (2020) proposed a novel community detection inspired by natural phenomena based on Matthew effect (CDME) that uses batch processing (Sun, Sun, et al., 2020). CDME considers a network as a social system, inspired by the Matthew effect in human society to simulate the process. This technique finds dynamically changing high-quality communities. A major benefit of CDME is the parameter-free. This makes the technique less parameter-sensitive and faster.

The local CDME calculates network vertex attractiveness and localization to help it manage vast networks. Considering networks as social systems, the CDME finds community structures using the number of common neighbors and connection strength, which determines the attractiveness of neighboring vertices in the network. Each vertex is assigned to a community, and then merges communities based on attractiveness. However,

CDME may still face challenges with parameter sensitivity, batch processing efficiency, bias towards larger communities, and the need for extensive validation to ensure robustness.

Another existing community detection inspired by natural phenomena is community detection based on fish school effect (CDFSE) (Y. Sun et al., 2022). It is one of the unique community detection techniques that is inspired by fish schools formation. CDFSE inspires networks as ecosystems and presents a dynamic model to show communities more intuitively. The CDFSE features good community detection, parameter-free operation, and scalability. However, it may face challenges with high time complexity and in adapting to dynamic network changes.

These community detection techniques have been widely studied and compared in the literature, showing various strengths and limitations depending on network size, structure, and dynamics (Rossetti et al., 2017; Javed et al., 2018; Akbar and Saritha, 2020; Chunaev, 2020; Jin et al., 2021; Su et al., 2024).

2.3 Process of Community Detection

Community detection is a fundamental task in network analysis which aims to identify groups of vertices that are more densely connected to each other than to the rest of the network. The process of community detection typically involves three key components: similarity measure, vertex attraction, and module attraction (Fortunato, 2010; Su et al., 2024). Figure 2.6 illustrates the process of community detection, beginning with the measurement of similarity, followed by assessment of vertex attraction, and culminating in the determination of module attraction to identify cohesive communities within a network. The effectiveness of the community detection relies on the seamless integration and

cooperation of these components. Together, these components guide the identification of meaningful communities within networks, ensuring that they remain stable and adaptive to changes over time.

Figure 2.6 Process in community detection

Similarity measures evaluate the strength of relationships between vertices based on their structural properties, helping to identify the degree of similarity between them. The similarity measure is crucial as it sets the stage by quantifying how similar vertices are, which directly influences vertex attraction. Vertex attraction determines how strongly individual vertices are drawn to potential communities, influencing their placement within these communities. This attraction depends on similarity, as vertices that are more alike will naturally exert a stronger pull towards each other. In turn, module attraction relies on the outcomes of vertex attraction, ensuring that the groups formed are cohesive and distinct enough to be recognized as separate communities. This dependency highlights that the strength and accuracy of the final community structure result from the combined effectiveness of each component, with each step building on the results of the previous one. Without this interdependence, the technique would struggle to detect communities that are both meaningful and well-defined.

2.3.1 Similarity Measure for Community Detection

As mentioned earlier, selecting an appropriate similarity measure plays a crucial role in community detection. It evaluates the likelihood of connections between vertices and assesses how closely related the vertices are within a network. Most similarity calculations can be categorized as local-based, global-based, or quasi-based functions. Typically, these methods are unsupervised. Each dynamic community detection approach uses its own technique to calculate the similarity measure and assess community quality, often relying on local structural information from the testing set.

The similarity measure is essential for accurately identifying and grouping similar vertices within networks. It enables community detection to analyze the structure of complex networks, leading to valuable insights in many fields, such as social network analysis, biology, and marketing. This section defines and describes seven different similarity measures, each focusing on various aspects of vertex relationships. The section begins with a brief introduction to each measure or index.

2.3.1.1 Common Neighbour Index

The Common Neighbors (*CN*) index was introduced by Liben-Nowell and Kleinberg in 2003. *CN* index is a local similarity measure used to estimate the likelihood of an edge existing between two vertices by analyzing their neighborhood structures. Specifically, the *CN* index calculates the number of shared neighbors between vertices u and v, as represented by Equation (2.1).

$$CN_{uv} = |\Gamma(u) \cap \Gamma(v)| \tag{2.1}$$

where $\Gamma(u)$ represents its set of neighbors of vertex u, respectively. The basic idea behind this measure is that two vertices u and v are more likely to be connected if they share many common neighbors, making it a fundamental concept in link prediction and network analysis (Liben-Nowell and Kleinberg, 2003).

2.3.1.2 Jaccard Index

The *Jaccard* index was proposed by Jaccard in 1901, and is defined as in Equation (2.2).

$$Jaccard_{uv} = \frac{|\Gamma(u) \cap \Gamma(v)|}{N(u,v)} \tag{2.2}$$

where $\Gamma(u)$ and $\Gamma(v)$ represent the sets linked to u and v, respectively. The numerator, $|\Gamma(u) \cap \Gamma(v)|$, quantifies the number of common elements shared by both sets, while the denominator, $N(u,v)$, accounts for the total unique elements across both sets where equals to $|\Gamma(u) \cup \Gamma(v)|$. The Jaccard index yields a value between 0 and 1, where a score of 0 indicates no similarity between the sets (i.e., no shared elements), and a score of 1 signifies that the sets are identical. This measure is particularly valuable in applications where assessing the degree of overlap or shared attributes is essential (Jaccard, 1901). The Jaccard index is frequently utilized in research, with slight variations in the normalization function, as cited by several authors such as Greene (2010), Nguyen et al. (2012), and Pereira et al. (2021), even though it may be named differently in some cases.

2.3.1.3 Salton Index

Another similarity measure, Salton index was proposed by G. Salton in 1975 and is represented by Equation (2.3).

$$Salton_{uv} = \frac{|\Gamma(u) \cap \Gamma(v)|}{\sqrt{D(u) \times D(v)}} \tag{2.3}$$

where $D(u) = |\Gamma(u)|$ denotes the degree of vertex u, and $D(v) = |\Gamma(v)|$ denotes the degree of vertex v. The *Salton* similarity index is a measure of similarity between two vertices in a network and is mathematically represented as the ratio of the number of common neighbors of the vertices to the product of their degrees. This index is also known as the cosine similarity, which is widely used in various fields, particularly in information retrieval and network analysis, to evaluate the strength of the connection between two vertices. The index provides a normalized measure, allowing for the comparison of vertex similarity within a network, regardless of the scale of their respective connections (Salton et al., 1975).

2.3.1.4 Sorensen Index

Sorensen index is used mainly for eco-logical community data. *Sorensen* index (also known as the Sorensen-Dice coefficient) is another similarity measure used to compare the similarity and diversity of sample sets. It was proposed by T. Sorensen in 1948, and is defined as in Equation (2.4).

$$Sorensen_{uv} = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u)| + |\Gamma(v)|} \tag{2.4}$$

where $\Gamma(u)$ represents its set of neighbors of vertex u, respectively. The *Sorensen* similarity measure is primarily utilized in ecological studies to quantify the similarity between two communities. The index is calculated by taking the ratio of the intersection of the two sets to the sum of their sizes, effectively measuring the proportion of shared elements between the two communities. This index is widely used because it provides a straightforward and interpretable measure of similarity, particularly in contexts where the goal is to assess the degree of overlap between two distinct groups or sets (Sorensen, 1948).

2.3.1.5 Leicht-Holme-Newman Index

Leicht-Holme-Newman Index (*LHN*) similarity measure was proposed by Leicht, Holme, and Newman in 2006, and assigns high similarity to vertex pairs that have many common neighbours compared not to the possible maximum, but to the expected number of such neighbour. It is defined as in Equation (2.5).

$$LHN_{uv} = \frac{|\Gamma(u) \cap \Gamma(v)|}{D(u) \times D(v)} \tag{2.5}$$

where $D(u) = |\Gamma(u)|$, and the denominator $D(u) \times D(v)$ adjust the similarity score based on the degree of vertices u and v, making it proportional to the expected number of common neighbors. The *LHN* is designed to assess the similarity between vertex pairs by considering the number of common neighbors they share. Unlike other measures, *LHN* does not simply compare the observed common neighbors to the possible maximum but rather to the expected number of such neighbors, providing a more refined similarity assessment (Leicht et al., 2006).

2.3.1.6 Adamic Adar Index

Adamic-Adar (*AA*) measure was proposed by Adamic and Adar in 2003 and is defined as in Equation (2.6).

$$AA_{uv} = \sum_{u \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log |D(u)|} \tag{2.6}$$

where $\sum$ is the summation symbol, and this is the term being summed for each common neighbor of vertex u. *AA* is used to quantify the similarity between two vertices in a network. It is the inverse of the logarithm of the degree of vertex u, where $D(u)$ is the degree (the number of connections) of vertex u. The degree is often denoted as $D(u) = |\Gamma(u)|$. The rationale behind the *AA* measure is that common neighbors with fewer connections (i.e., lower degree) contribute more to the similarity score, emphasizing the importance of less connected, more exclusive common neighbors in the similarity calculation (Adamic and Adar, 2003).

2.3.1.7 Resource Allocation Index

Resource Allocation (*RA*) is a more recent similarity measure introduced by Zhou et al. in 2009, as represented by the formula in Equation (2.7).

$$RA_{uv} = \sum_{u \in \Gamma(u) \cap \Gamma(v)} \frac{1}{|D(u)|} \tag{2.7}$$

where $D(u) = |\Gamma(u)|$ and the term $D(u)$ refers to the degree of a vertex u, which is the number of edges connected to it. RA_{uv} represents the resource allocation between two

vertices, u and v in a network. $\sum$ is the summation symbol, indicating the summation over all the inverse of the degree of each common neighbor (Zhou et al., 2009).

RA is similar to *AA* but inspired by resource flow. RA often performs compared to other indices in sparse network due to its resource flow interpretation. This approach highlights that neighbors with fewer connections have a greater impact on the similarity between vertices. It is particularly useful in network analysis, as it reveals structural similarities by analyzing shared connections between vertices. This method is highly effective and often surpasses other local methods in comparative studies. Despite being the simplest similarity functions, local methods are widely preferred for their efficiency and scalability.

2.3.2 Limitations of Similarity Measure in Community Detection

Although similarity measures such as *CN* , *Jaccard* , *Salton* , *Sorensen* , *LHN*, *AA*, and *RA* indices are widely used and recognized as important in community detection, there are still limitations and challenges. These measures primarily focus on the local structure of networks, analyzing direct connections between vertices or the overlap in their neighborhoods. While this local perspective can be effective in detecting tightly knit communities, it is often too narrow, overlooking the broader network topology and the influence of more distant vertices.

Stability issues also arise in similarity measures due to their sensitivity to small changes in the network's structure. For instance, the addition or removal of a single edge or vertex can significantly alter pairwise similarity scores, leading to inconsistent community

assignments. This instability is particularly problematic in dynamic networks, where frequent changes occur. In such cases, the community detection process can become unreliable, with results fluctuating as the network evolves over time. As shown in Figure 2.7, the trends of similarity measures across time reveal the growing adoption of some measures, with RA and AA showing considerable use in recent years. This information is adapted from Liu et al. (2017).

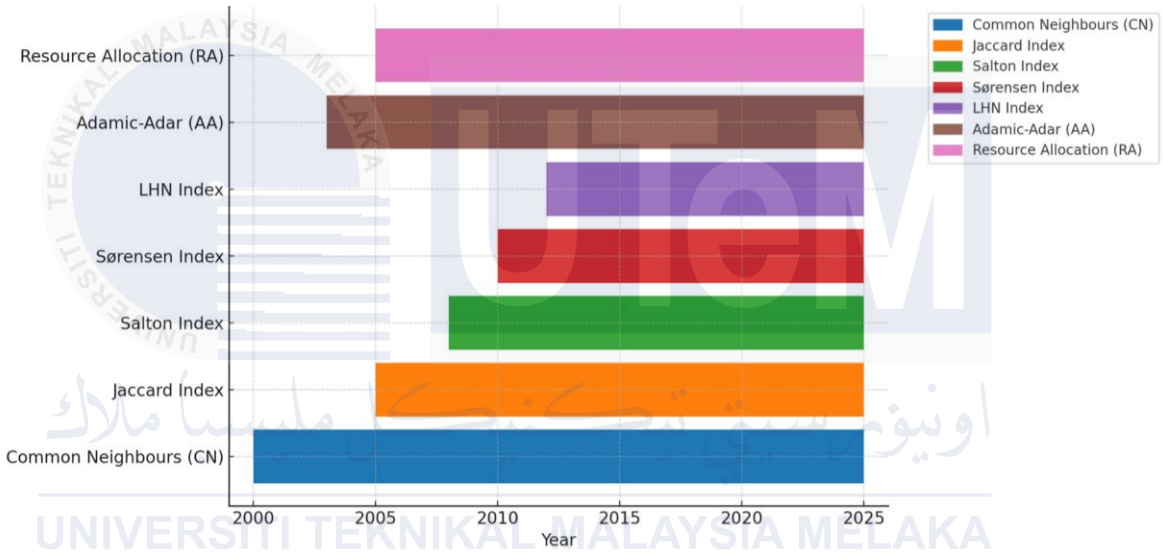


Figure 2.7 Usage timeline of similarity indices used in DCD (2000-2025)

For instance, the *CN* index simply counts the number of shared neighbors between two vertices, which can lead to biased results in networks with high-degree vertices, as these vertices naturally have more connections and hence more shared neighbors. The Jaccard and Sorensen indices, which normalize the number of common neighbors by the total number of neighbors, attempt to address this bias but can still struggle in networks with uneven degree distributions. Additionally, the Salton index, which is based on the cosine similarity of neighbor vectors, can be sensitive to small changes in vertex connectivity, potentially

leading to unstable community assignments. The *LHN*, AA, and *RA* indices try to incorporate more sophisticated weighting schemes to address these issues, but they often introduce additional computational complexity and may still fall short in capturing the true underlying community structure, especially in large or highly heterogeneous networks.

Another factor contributing to instability is the dependency on the choice of the similarity index or measure. Different similarity measures may yield vast different results, even when applied to the same network. This inconsistency complicates the reproducibility and reliability of the results, as the chosen index heavily influences the detected communities.

Furthermore, these similarity measures generally assume that all edges or connections have equal importance, which might not be true in real-world networks where some connections are stronger or more influential than others. Moreover, similarity measures often fail to maintain stability when applied to noisy or incomplete data. Real-world networks frequently contain missing or erroneous information, and similarity measures that rely on direct relationships between vertices may propagate these errors, leading to unstable community detection outcomes.

As a result, while these indices provide valuable insights and serve as foundational tools in community detection, they must be applied with an understanding of their limitations and in conjunction with other methods that can account for the complexities of network structures.

Based on previous study, FaceNet employs Kullback-Leibler divergence as its similarity measure, while IncNSA uses the *Salton* measure, DCDID utilizes the *Jaccard* index, and DCDME applies the *AA* index to calculate the similarity of the initial community

formations. Each of these indices has its own strengths and capabilities, yet challenges remain depending on the dataset or situation. In conclusion, based on these seven similarity measures, the partition of the vertices at the initial phase will be carried out.

2.3.3 Vertex Attraction for Community Detection

After completing the initial process of community detection using similarity measures, the next process is vertex attraction (VA). VA determines how strongly a vertex is pulled into a community based on similarity or connectivity. This step is pivotal in forming coherent communities and is often guided by heuristic or probabilistic approaches. Generally, the vertex attraction is proportional to both the degree of a vertex and its similarity with neighboring vertices, meaning that highly connected vertices with strong similarity exert greater influence within the community (Liben-Nowell and Kleinberg, 2003). The highest degree value will attract the similar vertices. The concept of vertex attraction can be expressed as shown in Equation (2.8).

$$VA_{i,j} = f\left(\deg_i, sim_{i,j}\right) \tag{2.8}$$

where, $\deg_i$ represents the degree of vertex i, and $sim_{i,j}$ represents the similarity between vertices i and j.

The VA formula can be found in studies that address link prediction and community detection in dynamic networks, especially where attraction between vertices is considered as a function of their similarity and connectivity. Vertex similarity is better used to compute weights for existing edges or enable the model to reflect both the current network topology and potential future connections. (Z. Sun et al., 2022).

In this study, the concept of VA is extended into a multi-level module attraction framework. Instead of measuring attraction only between individual vertices, the proposed technique also considers attraction between modules (communities). This helps the model capture both local and global relationships in the network, maintaining community stability as the structure changes over time. By combining similarity measures with module attraction, the technique improves the consistency of detected communities and reduces instability during network evolution.

To make the attraction process more adaptive, this research applies bird flocking behavior principles. Similar to how birds move together through cohesion, alignment, and separation, vertices in the network adjust their positions based on their connections and similarities. Cohesion pulls vertices toward the center of their community, alignment keeps them moving consistently with their neighbors, and separation avoids overlapping with other communities. These natural rules help the communities adapt smoothly to network changes and remain stable over time. Overall, the VA process in this thesis is a key part of the proposed dynamic community detection technique.

2.3.4 Module Attraction for Community Detection

Following the two processes in community detection, namely similarity measure and vertex attraction, the third process is module attraction (MA) (Z. Sun et al., 2022). In this phase, interactions between vertices are observed when similarity measures and vertex attraction are applied to identify communities within a complex network. The concept of module attraction in dynamic community detection plays a crucial role in forming

communities, with particular emphasis on the connectivity level between vertices. Various techniques have been developed to optimize this process, each employing different strategies to measure and enhance module attraction.

For instance, Lin et al. utilized a cost function as module attraction within the FaceNet, aiming to minimize the overall cost of connecting vertices in a way that preserves the structural integrity of the community (Lin et al., 2009). Similarly, Folino and Pizzuti applied a cost function in their DYNMOGA to balance the trade-offs between maintaining community cohesion and minimizing the cost associated with forming these communities (Folino and Pizzuti, 2014). The focus on cost functions in these techniques underscores the importance of reducing unnecessary connections while still preserving the essential structure of the community.

On the other hand, some techniques emphasize the modularity of the network as a key factor in module attraction. Nguyen et al. and Su et al. implemented the QCA and IncNSA, respectively, with both relying on modularity to assess the quality of the communities formed (Nguyen et al., 2011; Su et al., 2020). Modularity measures the strength of the division of a network into communities, with higher modularity indicating stronger and more distinct community structures.

Enhancing module attraction in dynamic community detection involves a focused consideration of the level of connectivity between vertices and the application of weighted multi-layer techniques. For example, Sun et al. proposed CDME and the community attraction is only first layer, $CA_{v \to c_i} = D_{c_i}(v)^2$, where $D_{c_i}(v)$ denotes the degree of connectivity from a vertex v to one community c_i, which portrays how close a vertex is to one community (Y. Sun et al., 2022). He introduced a more complex approach in the

DCDME by incorporating group attraction and a second layer of analysis to refine the detection of community boundaries. The group attraction is $GA_{u\to g_j} = D_{g_j}(u)^2 + \sum_{v\in N(u), v\in g_j} d_{g_j}(v)$, where $D_{g_j}(u)$ represents the degree of connectivity from vertex u to a group g_j, which describes the closeness of the vertex to the group.

Based on the connection between vertices and different groups, there are two classifications on whether the number of edges differs or is the same. v is a neighbor vertex of u, and $d_{g_j(v)}$ denotes the internal degree of vertex v in group g_j. The combination of level connectivity, and module attraction measures across various techniques highlights the diverse approaches researchers have taken to improve the accuracy and efficiency of dynamic community detection in networks. In the literature, there are differences of taxonomies to categorize the changes of module, or patterns of dynamic network.

Recent research shows that community structures in complex networks naturally emerge at the mesoscopic scale, which corresponds to a neighbourhood radius of approximately two to three levels of neighbour connectivity. At this structural depth, vertices begin to share overlapping neighbourhoods, common connectivity patterns, and stable propagation influence, making the second- to third-level neighbourhood the most informative region for detecting cohesive module structure. First-level and second-level neighbour connectivity, while useful for capturing local ties, are often too limited to reveal the broader structural cohesion of a community. These shallow neighbourhoods primarily capture immediate relationships and short-range clustering. Studies on multi-level neighbourhood aggregation show that relying only on these local levels often results in

unstable or ambiguous community assignments, especially for boundary vertices whose local connections overlap multiple modules (Lei et al., 2025).

In contrast, expanding neighbourhood connectivity beyond the third level tends to introduce structural noise. At four or more levels of neighbour expansion, vertices begin to incorporate information from vertices outside their true module. This broad expansion mixes signals from unrelated communities and reduces the discriminative ability of attraction measures. Research on structural propagation indicates that meaningful influence and structural consistency diminish significantly after the third-level neighbourhood, and further levels add noise rather than useful information (Shetty et al., 2025).

Thus, the third-level neighbourhood forms a theoretically justified boundary for capturing meaningful module attraction. It provides access to sufficient structural context to reflect mesoscopic community cohesion while remaining constrained enough to avoid contamination from distant or unrelated parts of the network. This theoretical foundation supports the evaluation and use of module attraction up to the third level in this thesis. In Chapter 3.2.2.2, the formal definitions of module analysis component are provided in more detail.

2.3.5 Limitation of Module Attraction in Community Detection

The summary of similarity measures and module attraction used in well-known existing techniques is presented in Table 2.3. Module attraction focuses on aggregating smaller communities or modules into larger, more cohesive structures. Some of the limitations include modularity resolution, cohesiveness constraints, and information loss, as highlighted in various studies. Stability in module attraction processes is often compromised

by the resolution limit problem. Modularity optimization for example, tends to merge or split communities inconsistently when the network size or resolution parameters change. This instability can result in different community structures being detected for the same network under slightly altered conditions (Fortunato, 2010).

Furthermore, another source of instability is cohesiveness constraints or the hierarchical nature of module attraction. The community structure is sometimes compromised when connections across different levels of the network are inadequately considered, leading to incomplete or imprecise community formation. As smaller modules are aggregated into larger ones, minor inconsistencies or inaccuracies in earlier stages can propagate and amplify, leading to significantly different outcomes. This issue is particularly pronounced in multi-leveled or weighted networks.

Finally, information loss arises when simplifying assumptions in the module attraction process failed to capture the nuances of complex networks. Most existing dynamic community detection rely on similarity measures such as Common Neighbors, Jaccard Coefficient, or Preferential Attachment, which mainly capture static structural proximity between connected vertices. These measures perform well in identifying local connections but are less effective in maintaining temporal continuity when the network structure changes over time. More advanced indices like Adamic-Adar and Resource Allocation improve sensitivity by considering vertex degree influence, yet still lack the ability to represent how community relationships evolve. These factors collectively emphasize the need for continuous innovation in the field to overcome inherent limitations (Lancichinetti and Fortunato, 2009).

Table 2.3 Comparison of similarity measures and module attractions used in existing DCD

Author	Year	Name of DCD	Similarity measure used	Equation of similarity used	Module attraction used	Equation of module attraction used
Lin et al.	2009	FaceNet	Kullback-Leibler divergence	$KL(P \| Q) = \sum P(i) \times \log\left(\frac{Q(i)}{P(i)}\right)$	Cost function	$\cos t = \alpha \cdot D\left(W \| X \wedge X^T\right) + \left(1-\alpha\right) \cdot D\left(Y \| X \wedge\right)$
Nguyen et al.	2011	QCA	Modularity	$Q = \frac{1}{2m} \sum \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(C_i, C_j)$	Modularity	$Q = \sum \frac{m}{M} - \frac{d_C^2}{4M^2}$
Folino & Pizzuti	2014	DYNMOGA	Modularity	$Q = \sum_{s=1}^{k} \left[\frac{l_s}{m} - \left(\frac{d_s}{2m} \right)^2 \right]$	Cost function	$\cos t = \alpha \cdot SC + \left(1-\alpha\right) \cdot TC$
Agarwal et al.	2018	DyPerm	Permanence	$Perm(v) = \left[\frac{I(v)}{E_{\max}(v)} \times \frac{1}{d(v)} \right] - \left[1 - C_{in}(v)\right]$	Cohesiveness	$C_{in}(v) = \frac{E_{neig}(v)}{\left(\frac{I(v)}{2} \right)}$
Su et al.	2020	IncNSA	Salton	$Salton_{uv} = \frac{\lvert \Gamma(u) \cap \Gamma(v) \rvert}{\sqrt{k(u) \times k(v)}}$	Modularity	$Q = \sum_{i}^{K} \left(e_{ii} - a_i^2 \right)$
Sun et al.	2020	DCDID	Jaccard	$Jaccard_{uv} = \frac{\lvert \Gamma(u) \cap \Gamma(v) \rvert}{N(u,v)}$	Information loss	$I = \frac{AvgSim}{AvgDeg} f(I_u - I_v) * (1 - Jaccard_{uv})$
Sun et al.	2022	DCDME	Adamic Adar	$AA_{uv} = \sum_{u \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log \lvert k(u) \rvert}$	Group attraction, Second layer	$GA = D(u)^2 + \sum d(v)$

2.4 Overview of Dynamic Community Detection

Traditional community detection, often referred to as static community detection, is a fundamental task in network analysis. It focuses on identifying communities or modules within a network based on the structure of connections at a single point in time. However, in real-world applications, networks are rarely static; they tend to evolve over time as new data accumulate, connections form or dissolve, and the overall network structure shifts (Dakiche et al., 2019). This dynamic nature of data requires adaptive approaches that can continuously monitor and analyze the evolving network structure, identifying how communities change, merge, or split over time. Dynamic community detection aims to address these challenges, offering a more realistic and accurate understanding of network behavior by capturing temporal changes and adapting to the growth of data. This approach is essential in various domains, including social media analysis, biological networks, and recommendation systems, where understanding temporal dynamics can provide valuable insights into the evolution of relationships and influence within complex networks.

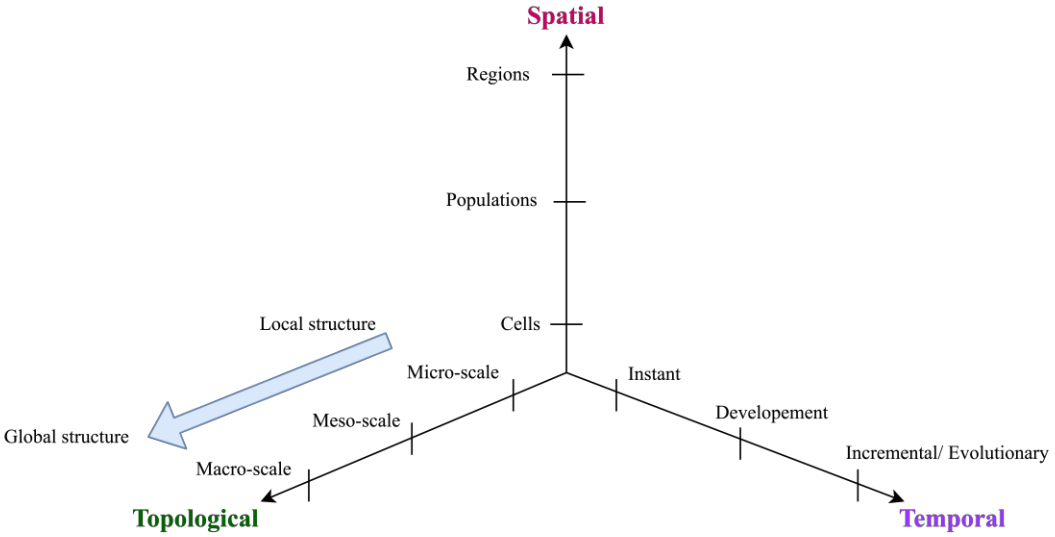


Figure 2.8 Dimensions of community structure in complex networks

Dimensions of community structure in complex networks with three components illustrated in Figure 2.8 are topological, spatial, and temporal. The topological dimension focuses on the structure of the network, analyzing the arrangement of vertices and edges, community formation, and connectivity. The topological dimension of community structure in complex networks across multiple scales, highlights the progression from local structures at the micro-scale, through intermediate or meso-scale groupings, to the overarching global structure at the macro-scale. This hierarchical approach allows for the analysis of community formation at different levels of granularity, offering insights into how small, localized interactions contribute to larger, network-wide patterns and dynamics.

In addition, the spatial dimension, depicted vertically in Figure 2.8, illustrates how communities can be organized across different physical or conceptual spaces, ranging from individual cells and populations at smaller scales to larger regions. It emphasizes the importance of spatial considerations in understanding the distribution and interaction of communities within complex networks. The spatial dimension takes into account the physical or geometric space that influences vertex connections, which is crucial in networks where proximity affects interactions.

The temporal dimension, shown horizontally in Figure 2.8, captures the evolution of community structures over time, from instantaneous time steps to gradual developmental changes, and finally to incremental or evolutionary adaptations, reflecting the dynamic nature of communities in response to changing conditions within the network. The temporal dimension addresses the dynamic changes within the network over time, including the evolution of connections and communities. These dimensions collectively offer a comprehensive understanding of how networks are structured, influenced by spatial factors,

and evolve over time. Together, these three dimensions: topological, spatial, and temporal are integral to dynamic community detection, as they provide a comprehensive framework for analyzing how communities form, interact, and evolve over time within complex networks.

In complex networks, DCD have been specifically developed to uncover the intricate structures of communities and understand their evolution over time. These techniques are particularly effective in dynamic networks where both the structure and interactions among vertices frequently change. In recent years, many techniques have been introduced to capture the dynamics of community structures, reflecting the growing interest in understanding how social phenomena evolve. The study of dynamic community detection has gained increasing attention as it holds significant potential for revealing how networks and the social dynamics they represent transform over time.

Over the past ten years, numerous papers addressing the temporal evolution of communities were published, utilizing various approaches. Figure 2.9 shows that the development of this field has been recognized through several comprehensive review papers, which highlight the most widely adopted approaches based on insights from previous research on community detection and evolution in temporal networks.

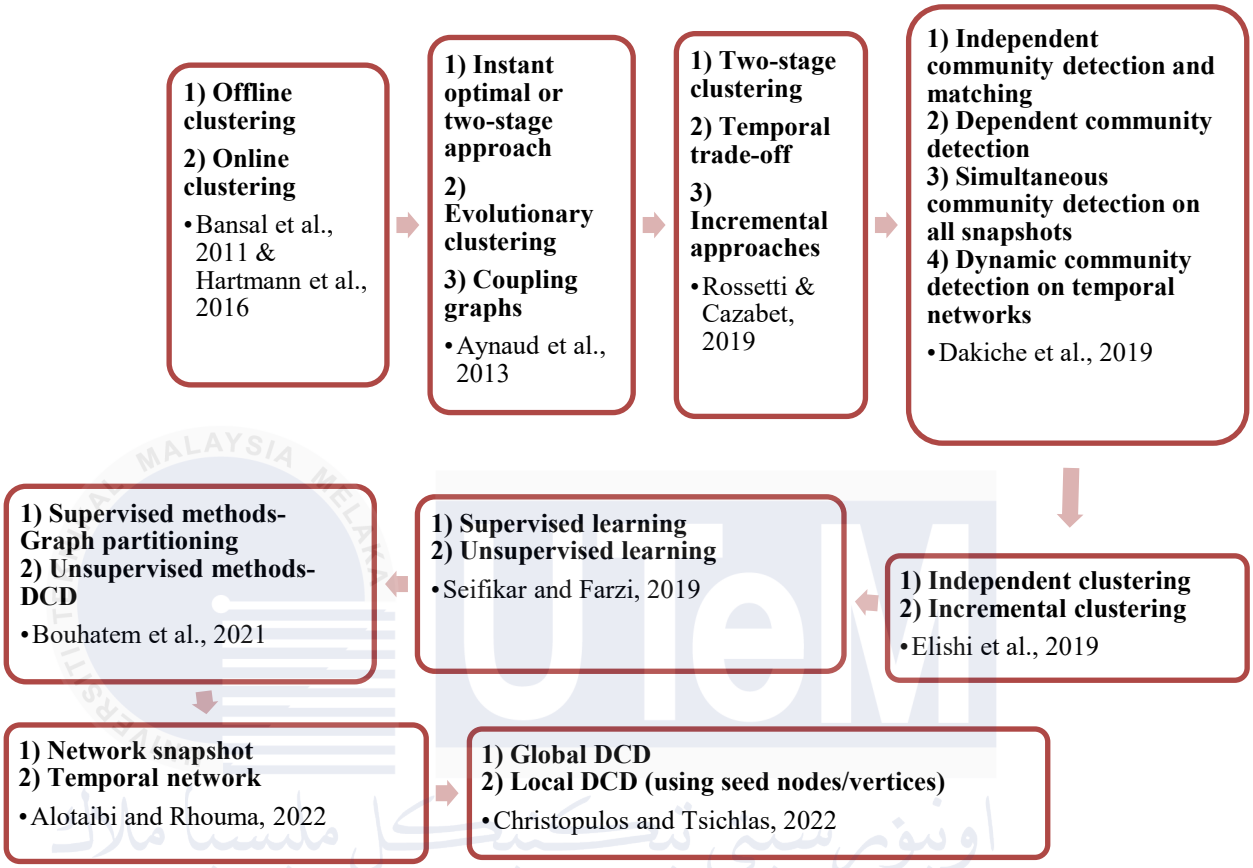


Figure 2.9 Evolution and classification of dynamic community detection

The existing classification schemes in this field originated from the ideas of Bansal, Bhowmick and Paymal, 2011, as well as Hartmann, Kappes, and Wagner, 2016, who identified two primary approaches: offline and online clustering. This classification has evolved with contributions from Aynaud et al., 2013, Rossetti and Cazabet, 2018, and Dakiche et al., 2019, who introduced additional perspectives.

Aynaud et al. proposed three categories for detecting communities: two-stage approaches, evolutionary clustering, and coupling graphs. Rossetti and Cazabet identified three themes in dynamic community detection: instant optimal or two-stage approaches, temporal trade-off, and cross-time or incremental approaches. Furthermore, Dakiche et al.

classified studies on tracking community detection in dynamic social networks into four main approaches: independent community detection and matching, dependent community detection, simultaneous community detection on all snapshots, and dynamic community detection on temporal networks (Dakiche et al., 2019).

Table 2.4 provides a comprehensive overview of the evolution and various approaches to dynamic community detection as explored in different studies. It summarizes key research contributions from authors across several countries, highlighting the diversity of approaches used to address the challenges in dynamic community detection.

Table 2.4 Summary of evolution and approaches of dynamic community detection

No.	Authors (Year)	Country	Journal	Approaches
1	Bansal, Bhowmick and Paymal (2011)	USA	Communications in Computer and Information Science (Conf.Paper)	1) Offline clustering 2) Online clustering
2	Hartmann, Kappes, and Wagner (2016)	Germany	Lecture Notes in Computer Science	
3	Giatsoglou and Vakali (2013)	Greece	IEEE Internet Computing	1)Sequential mapping 2)Temporal smoothing or evolutionary clustering) 3)Milestone detection 4)Incremental adaptation
4	Aynaud et al. (2013)	France	Modeling and Simulation in Science, Engineering and Technology	1)Two-stage approaches 2)Evolutionary clustering 3)Coupling graphs
5	Liu et al. (2018)	China	Intelligent Data Analysis	1) Dynamic clustering / Evolutionary clustering 2) Objective function optimization 3) Representative vertex (community) detection 4) Dynamic probability modelling

No.	Authors (Year)	Country	Journal	Approaches
6	Rossetti and Cazabet (2018)	France	ACM Computing Surveys	1) Instant optimal / Two-stage approach 2) Temporal trade-off 3) Cross-time/ Incremental approaches
7	Dakiche et al. (2019)	Algeria	Information Processing and Management	1) Independent Community Detection and Matching 2) Dependent Community Detection 3) Simultaneous Community Detection on All Snapshots 4) Dynamic Community Detection on Temporal Networks
8	Elhishi *et al*. (2019)	Egypt	Behaviour and information technology	1) Independent approach 2) Incremental approach / evolutionary clustering
9	Seifikar and Farzi (2019)	Iran	Journal of Information Science	1) Supervised learning 2) Unsupervised learning
10	Taha (2020)	United Arab Emirates	IEEE Access	1) Static clustering 2) Dynamic clustering
11	Bouhatem et al. (2021)	Algeria	Acta Universitatis Sapientiae, Informatica	1) Supervised methods-Graph partitioning methods 2) Unsupervised methods-DCD • two-stage approaches • dependent approaches • incremental
12	Alotaibi and Rhouma (2022)	Saudi Arabia	Journal of King Saud University - Computer and Information Sciences	Dynamic network: 1) Network snapshot (Independent, dependent) 2) Temporal network (Disjoint, Overlapping communities)
13	Christopoulos and Tsichlas (2022)	Greece	IFIP Advances in Information and Communication Technology	1) Global DCD • From stratch and Match • Dependent/ Temporal Trade off • Simultaneous/ Offline • Online DCD 2) Local DCD • Using seed vertices

The table categorizes approaches into offline and online clustering, sequential mapping, evolutionary clustering, independent and dependent approaches, and more. This diversity reflects the complexity of dynamic networks and the ongoing efforts to develop more accurate and efficient techniques for detecting communities within these networks. The table underscores the importance of adapting approaches to specific scenarios, such as independent or dependent community detection, which are critical for advancing the field of dynamic community detection.

From the comparative study of all above, the best explanation for tracking the community detection in a dynamic network is still subjective. This thesis follows the thematic classifications proposed by researcher Elishi et al. and Bouhatem et al. that proposed dynamic community detection approaches can be broadly classified into two main categories: independent and dependent community detection (Elhishi et al., 2019 and Bouhatem et al., 2021). This thesis adopts their thematic classifications because their approach provides a clear framework for distinguishing between independent and dependent community detection techniques, enabling a systematic and structured analysis of how different techniques address the complexities of community detection in dynamic networks.

As illustrated before in Figure 2.1, dynamic community detection is divided into two. Independent and dependent techniques. Independent community detection techniques analyze each network snapshot separately, while dependent community detection techniques consider the temporal continuity between snapshots. Dependent techniques can be further divided into evolutionary and incremental approaches, some of which are inspired by natural phenomena.

2.4.1 Independent Community Detection

Independent community detection is also known as a two-stage clustering. First, applying static community detection methods to dynamic cases yielding this first approach and then, dividing the network's progression into numerous time steps and assign communities to each. Each time steps detect communities separately and match them with the ones from the previous stage based on similarity or distance measure. The one using a similarity metrics is for instance Jaccard (Dakiche et al., 2019; Kadkhoda Mohammadmosaferi and Naderi, 2020). The advantage of this dynamic community detection is it is directly used to the dataset without changes. However, the result of dynamic community detection is unstable which is one of its disadvantages. This technique can lead to inconsistencies in the detected communities across consecutive time steps, as it does not account for prior information.

2.4.2 Dependent Community Detection

Dependent community detection eliminates the need to match communities, streamlining the community identification process. Instead, this approach focuses on identifying communities in the present time period based on those detected in prior periods, ensuring continuity and consistency in the analysis. In dynamic community detection, two main approaches are employed: the cost function method and the direct method (Pereira, Lopes and Louçã, 2021). The cost function method is typically aligned with the evolutionary approach, where the focus is on maintaining smooth transitions in community structures over time. This method works by optimizing a cost function that minimizes disruptions between

communities at consecutive time steps, ensuring stability and coherence as the network evolves. On the other hand, the direct method is associated with the incremental approach. This method directly utilizes information from the previous time step to detect communities in the current time step, allowing for quick adaptation to changes in the network. The direct method is particularly effective in scenarios where community structures may change abruptly or irregularly, making it a more flexible and responsive approach to dynamic community detection.

The differences between incremental and evolutionary approach in dynamic community detection are shown in Table 2.5. Several dynamic community detections have been proposed from several points of view and have their own benefits and drawbacks, such as prior information of parameter, high time complexity, and accuracy of detection decrease with respect to time step. However, the divergence from the static techniques is again a potential quality concern for incremental approach and needs to be evaluated, which is an efficiency of identifying community that still has limitation in terms of stability. This study emphasizes the incremental approach, which enables a more adaptable and responsive comprehension of community detection over time. This offers crucial insights into dynamic networks where flexibility is essential.

Table 2.5 Comparison aspects between incremental and evolutionary approaches in dependent dynamic community detection

Aspects	Incremental Approach	Evolutionary Approach
Definition	Updates communities based on network changes without full recomputation	Considers the entire history of the network to detect communities
Data Processing	Process new data or changes as they arrive	Analyzes the network over multiple time steps
Time complexity	Generally lower, as it only processes changes	Can be higher, as it often involves processing the entire history
Memory Usage	Usually lower, as it doesn't need to store full historical data	Higher, as it needs to store and process historical data
Adaptability	Quick adapts to recent changes	Considers long-term trends and patterns
Historical Context	May not fully consider long-term historical patterns	Incorporates full historical context in the community detection
Stability	Can be less stable due to sensitivity to recent changes	Generally more stable as it considers long-term patterns
Applicability	Suitable for streaming data and real-time analysis	Better for understanding long-term community detection

This thesis emphasizes enhancing the stability aspect of the incremental approach in dependent dynamic community detection. While the incremental approach offers lower time complexity and memory usage by processing changes as they arrive, it can be less stable due to its sensitivity to recent changes. By focusing on improving stability, this work aimed to retain the adaptability and efficiency of the incremental approach while mitigating fluctuations that may arise from short-term variations.

2.4.3 Incremental Dynamic Community Detection

The incremental approach in dynamic community detection is designed to utilize information from previous time steps to influence the identification of communities in subsequent ones, ensuring temporal consistency and smooth transitions in dynamic community detection results over time. Incremental approach uses global community discovery to find changed structures in each network time step, reducing time complexity (Christopoulos and Tsichlas, 2022). However, this approach is inherently limited by its inability to detect the evolution of communities within dynamic social networks that have explicitly defined community structures. This limitation arises because the incremental approach is more effective in networks where community structures are relatively stable over time, and it struggles when faced with scenarios involving abrupt or significant changes between time steps. Consequently, the approach might fail to capture the dynamic nature of evolving communities, potentially leading to inaccurate or outdated community structures. Recent studies, such as those reviewed by some researchers, underscore these stability challenges and suggest that while incremental community detection reduce time complexity, they may not be suitable for all types of dynamic networks (Zarayeneh and Kalyanaraman, 2021; Qu et al., 2022; Z. Sun et al., 2022; Chinichian et al., 2023).

In the followings, the current work on incremental community detection is reviewed in detail, and if possible, their limitation is explained. Table 2.6 describes several dynamic community detection with categories of approaches, strengths, weaknesses, and similarity measure used. The dynamic community detection techniques involved are FacetNet (Lin et al., 2009), QCA (Nguyen et al., 2011), DYNMOGA (Folino and Pizzuti, 2014), DyPerm

(Agarwal et al., 2018), IncNSA (Su et al., 2020), DCDID (Sun, Sheng, et al., 2020), and DCDME (Z. Sun et al., 2022).

Table 2.6 List of several existing dependent dynamic community detection

Author	Year	Name of DCD	Approach	Strength	Weakness	Similarity Measure Used
Lin et al.	2009	FacetNet	Evolutionary-Cost function	Accurate modeling of probability distributions	High computational cost	Kullback-Leibler divergence
Nguyen et al.	2011	QCA	Incremental-Direct	High accuracy in detecting communities with clear boundaries	Limited performance in networks with overlapping communities	Modularity
Folino et al.	2014	DYNMOGA	Evolutionary-Multiobjective	Balances multiple objectives for community detection, effective in dynamic environments	Computational complexity due to multi-objective nature	None
Agarwal et al.	2018	DyPerm	Incremental-Direct	Good at capturing stable communities over time	Sensitive to noise in the data	Permanence
Su et al.	2020	IncNSA	Incremental-Direct	Effective in networks with balanced vertex degrees	Limited scalability with large networks	Salton
Sun et al.	2020	DCDID	Incremental-Direct	Robust in networks with high similarity between communities	May struggle with dynamic or rapidly changing networks	Jaccard
Sun et al.	2022	DCDME	Incremental-Direct	Strong performance in detecting small or sparse communities	Potentially less effective in highly dense networks	Adamic Adar

Each approach has distinct advantages, such as FacetNet's accurate modeling of probability distributions, QCA's high accuracy in detecting clear community boundaries, and DyPerm's ability to capture stable communities over time. However, these approaches also

face limitations in which FacetNet has a high computational cost, QCA performs poorly in networks with overlapping communities, and DyPerm is sensitive to noise in the data. DYNMOGA utilizes multiobjective optimization to handle dynamic environments but come with increased computational complexity. Recent dynamic community detections, such as IncNSA, DCDID, and DCDME, focus on robustness and effectiveness in networks with varying degrees of density and similarity but may struggle with scalability or rapid changes in network structure. The table underscores the diversity in DCD methods and the trade-offs between accuracy, stability, and computational efficiency in dynamic community detection.

2.4.3.1 Framework for Analyzing Communities and EvoluTions in dynamic NETworks

Framework for Analyzing Communities and EvoluTions in dynamic NETworks (FacetNet) is a popular technique that proposes a hybrid approach that uses non-negative matrix decomposition and cost function optimisation for dynamic community discovery. FaceNet adapted an evolutionary clustering to study the dynamic network, where they use snapshot cost (SC), not snapshot quality. The SC at certain time is calculated as Kullback-Leibler (KL) divergence between the discovered community structure and the graph observed at this snapshot. Similarly, the temporal cost (TC) is defined as the KL-divergence between the communities discovered at time i-1 and i. The optimization problem is then to find the best community structure at time step i that minimizes the total cost. KL-divergence is used to measure how one probability distribution diverges from a second, expected probability distribution. However, one of its drawbacks is FacetNet's requirement for parameter settings, such as the quantity of communities in a network (Lin et al., 2009).

2.4.3.2 Quick Community Adaptation

Quick Community Adaptation (QCA) is a modularity optimization technique that adapts to network structure and information changes at the previous time step to find new communities (Nguyen et al., 2011). QCA is designed for dynamic online social networks with changing community structures. This adaptive modularity-based technique tracks community detection in these networks. QCA is very good at analyzing big and changing social networks because it is flexible and doesn't need many computer resources. The technique has been successfully used on real-life social networks, such as the Enron, arXiv, and Facebook networks. QCA has also shown that it can be useful in real life by being used for things such as socially aware message sharing in Mobile Ad Hoc Networks (MANETs) and stopping the spread of worms in Online Social Networks (OSNs). However, one drawback of the QCA is it often misses many structural properties of the communities, such as the number of vertices in the overlapping region and the memberships of the vertices. Another potential drawback of QCA is that it may not scale well with very large networks, leading to increased computational complexity and longer processing times.

2.4.3.3 Dynamic Multi-Objective Genetic Algorithms

Dynamic Multi-Objective Genetic Algorithms (DYNMOGA) is a well-known technique of evolutionary approach that finds communities in dynamic networks using a multi-objective optimisation based on genetic. DYNMOGA needs manual parameter specification to find the optimum partitions for dynamic community detection as a multi-objective temporal smoothness issue (Folino and Pizzuti, 2014). The purpose is to maximise

cluster accuracy with respect to current data and minimise clustering drift between time steps. This technique requires a user preference argument for time step or temporal quality. However, it may require parameter tuning and can have high spatial and temporal complexity, which increases the search space and computational demand that are challenging to detect communities in networks with rapidly changing dynamics.

2.4.3.4 Dynamic community Detection by maximizing Permanence

Dynamic community Detection by maximizing Permanence (DyPerm) is a dynamic community detection method that optimizes a novel community scoring metric called permanence (Agarwal et al., 2018). It incrementally modifies the community structure by updating those communities where the editing of vertices and edges has been performed, keeping the rest of the network unchanged. This approach leads to permanence maximization in dynamic networks, which in turn decreases the computational complexity drastically. However, a potential drawback of DyPerm is it requires the community structure of the initial time step, which can be obtained by running a static community detection and knows the ground-truth community structure. This requirement could limit its applicability in scenarios where such initial community structure is not readily available or accurate.

2.4.3.5 Incremental Node Similarity Algorithm

Incremental Node Similarity Algorithm (IncNSA) is an incremental approach that identifies dynamic communities through the examination of node (vertex) variations. IncNSA detects high-quality community structures in time-evolving networks. IncNSA only

analyses the community affiliations of partial vertices, which are either new or active from the previous time step, as networks grow. First, identify active vertices that need to be reassigned owing to community affiliation changes. Subgraphs for these vertices are created in the second step to create prototype communities. Third-phase optimization of primary communities yields the final output. This approach consistently detects good community structure from time step graphs, exceeding competitors. However, it may not efficiently handle the community affiliations of all vertices, because it focuses only on newborn vertices or certain active vertices from the previous network time step (Su et al., 2020).

2.4.3.6 Dynamic Community Detection based on Information Dynamic

Dynamic Community Detection based on Information Dynamic (DCDID) uses an information dynamics model to simulate vertex communication and exploits batch processing to incrementally identify communities in dynamic networks (Sun, Sheng, et al., 2020). Community detection can be improved by filtering out unmodified subgraphs. However, the DCDID's drawbacks are parameter tuning, high time complexity, and diminishing detection accuracy as time steps rise. As dynamic network changes get more sophisticated, the accuracy decreases. Then, it may suffer from high computational costs.

2.4.3.7 Dynamic Community Detection based on Matthew Effect

Dynamic Community Detection based on Matthew Effect (DCDME) is an incremental dynamic community detection inspired from natural phenomena (Z. Sun et al., 2022). A sociological concept called the Matthew effect (the rich get richer and the poor get

poorer) forms the basis of this technique. Vertices in a network gain more connections over time. Communities are difficult to discover in dynamic networks, which are continually changing.

Benefits of DCDME are parameter-free, and scales well. The DCDME outperforms various state-of-the-art algorithms in synthetic and real-world dynamic networks. However, DCDME is unsuitable for community detection on dynamic networks with a low clustering coefficient, which is an avenue for future improvement. In the context of dynamic networks, a low clustering coefficient might suggest that the network changes too rapidly for stable communities to form, or that the network's structure is decentralized, making it challenging for dynamic community detection. DCDME might struggle with variability and complexity and potentially leading to less stable community detection.

In conclusion, the discussion above is about the strengths, weaknesses and applicability. Most incremental dynamic community detection techniques adopt the fine-grained processing technique, which processes an event when an event is generated. Similar to this thesis, these techniques begin by analyzing the static communities detected at each time step using a community detection. Each of these approaches employs different similarity measures and strategies to track the evolution of communities over time, i.e., across time steps.

2.4.4 Inspiration from Natural Phenomena

In the context of dynamic networks and community detection, a biological system can be viewed as a complex network of interactions. Dynamic community detection in complex networks has been significantly influenced by natural phenomena, leveraging the

self-organizing and adaptive behaviors observed in nature to develop more effective methods for identifying community structures. A prominent example is the application of swarm intelligence principles, inspired by the collective behavior of animals such as bees, swarms, birds, fish, and insects. In these natural systems, individuals navigate their environment by following simple behavioral rules: matching the velocity of their neighbors, staying close to them, and avoiding collisions. Despite the simplicity of these rules, the group exhibits complex, adaptive, and coherent behavior without centralized control.

One specific example of natural law applied to dynamic networks is the bird flock effect model, which draws inspiration from the coordinated movements of birds in a flock. This model leverages characteristics of bird flocking, such as self-organization, collective decision-making, and adaptive behavior, to represent how entities in a network interact and influence each other (Beauchamp, 2021). From an ecosystem perspective, bird flocks demonstrate emergent behavior where individual birds follow simple rules based on local interactions, leading to complex global patterns. This phenomenon is analogous to dynamic community detection in networks, where communities emerge based on the interactions between vertices rather than predefined structures. This bio-inspired approach enhances both the efficiency and accuracy of community detection in evolving networks (Liu and Qiu, 2019).

Other examples include particle swarm optimization (PSO) and ant colony optimization (ACO), which have been adapted for community detection tasks. These techniques simulate how network vertices explore and adjust their connections to form cohesive communities by mimicking the collective movement and communication observed among individual particles or ants. For instance, the Multi-Objective Particle Swarm

Optimization (PSO) technique applies a modified version of PSO to detect communities by optimizing multiple objectives simultaneously. In this technique, each "particle" represents a potential solution, exploring the search space by adjusting its position based on its experiences and those of its neighbors. This approach effectively captures the dynamic and multi-objective nature of community detection, allowing the identification of well-defined and overlapping communities in complex networks (Rahimi et al., 2018).

Additionally, the parallel multi-objective evolutionary optimization technique (PMOEO-DCD), developed by Shen et al. (2022), incorporates principles from particle swarm optimization to dynamically detect communities. The Multi-Objective Particle Swarm Optimization Based on Network Embedding (MODPSO or NE-PSO) by X. Liu et al. (2020) also uses Salton or cosine similarity within a low-dimensional vector space to measure connections between vertices, further improving detection accuracy (X. Liu et al., 2020).

In a similar vein, Liu et al. (2022) introduced the Evolutionary Ant Colony Optimization (EACO) technique for detecting dynamic communities in vehicle movement networks. This technique emulates the behavior of ants seeking paths between their colony and food sources, using pheromone trails to adaptively identify communities in evolving networks. EACO employs the Jaccard index to effectively track community detection within these dynamic networks.

Evolutionary approaches, which simulate natural selection and genetic evolution, also provide significant inspiration for community detection. Techniques such as genetic algorithms (GA) utilize crossover and mutation operations to evolve solutions over time, effectively capturing the dynamic nature of communities as networks change. For example,

the Multi-Objective Pigeon-Inspired Optimization (MOPIO) technique, designed for community detection, uses negative ratio association (NRA) and ratio cut (RC) as objective functions. MOPIO employs genetic operators to redefine pigeon representation and update them during optimization, with NRA and RC calculations for each pigeon. Through Pareto sorting, non-dominated solutions are selected for crossover, while a leader selection strategy determines the final result from the optimal solution set (Shang et al., 2020).

Another relevant dynamic community detection is DYNMOGA, which uses principles of natural selection and genetic evolution—such as crossover and mutation—to evolve solutions for community detection in dynamic networks (Folino and Pizzuti, 2014). This, detailed further in Section 2.3.3.3, integrates with a Multi-Objective Genetic Algorithm with an Immigrant's scheme to reuse prior information and local search strategies to adapt to network changes. This combination aims to maintain solution quality while reducing computational time in evolving networks (Panizo-LLedot, Bello-Orgaz and Camacho, 2020).

Finally, this study was inspired by bird flock behavior, as demonstrated in the Sfloscan proposed by Bellaachia et al. in 2011. This bio-inspired technique detects communities in dynamic social networks by leveraging bird flocking behavior to group similar social network members into communities. The technique calculates similarity using Euclidean distance, forming communities within the spatial domain where each agent resides (Bellaachia and Bari, 2011). The fundamental concept of the bird flock effect is based on three primary rules of bird flocking: separation, alignment, and cohesion (Reynolds, 1987; Lijcklama à Nijeholt, 2020).

This thesis selected the bird flock effect for dynamic community detection because its adaptive and decentralized nature closely aligns with the characteristics of evolving networks, where relationships continuously form and dissolve. Additionally, bird flocking principles provide a natural framework for modeling local interactions that yield coherent global structures, making it highly suitable for detecting communities in dynamic settings (Beauchamp, 2021). Despite its potential, few studies have applied the bird flock effect in network analysis, highlighting an opportunity to explore its unique advantages further.

2.4.5 Limitations in Existing Dynamic Community Detection

Existing dynamic community detection techniques, while innovative and effective in specific scenarios, face several limitations that hinder their broader applicability and efficiency. FacetNet accurately models probability distributions using an evolutionary cost function but suffers from high computational cost, limiting its usability in large-scale networks (Lin et al., 2009). QCA demonstrates high accuracy in detecting communities with clear boundaries but performs poorly in networks where overlapping communities are prevalent (Nguyen et al., 2011). DYNMOGA with its multi-objective evolutionary approach, effectively balances multiple objectives in dynamic environments but encounters significant computational complexity due to its multi-objective nature (Folino et al., 2014).

DyPerm excels at capturing stable communities over time but is sensitive to noise in data, which can distort detection accuracy (Agarwal et al., 2018). Incremental approach like IncNSA and DCDID are effective in networks with balanced vertex degrees and high similarity between communities, respectively, but struggle with scalability in large networks and with rapidly changing network dynamics (Su et al., 2020; Sun et al., 2020). Finally,

DCDME shows strong performance in detecting small or sparse communities but is less effective in highly dense networks (Sun et al., 2022). These challenges highlight the need for further advancements to address computational efficiency, adaptability to overlapping or rapidly evolving structures, and scalability to larger and more complex network environments.

Based on the identified limitations, a key conclusion is that improving the stability metric in dynamic community detection requires refining similarity measures to balance sensitivity and robustness. In conclusion, the limitations of existing DCD techniques reveal a clear need for more efficient, scalable, and stable approaches. High computational costs, sensitivity to noise, and challenges in handling overlapping or rapidly evolving communities hinder the performance of current techniques, particularly in large-scale or complex networks. While incremental and evolutionary approaches offer valuable strengths, their weaknesses, such as limited scalability, computational complexity, and reduced effectiveness in dynamic or dense environments, underscore the necessity for novel techniques. Future research must focus on developing stable techniques capable of balancing accuracy, computational efficiency, and adaptability to diverse network structures and dynamics.

2.5 Behavioural and Event Used in Network Structure

In complex networks, dynamic community detection has been developed specifically to reveal the intricacies of community structures and comprehend their temporal evolution. The effectiveness of each approach is through optimization of the objective function. The

behavioural and role analysis about the changes on network can be easily understood from previous study in Table 2.7.

Community events were first proposed by Palla et al. in 2007 and, since then, there seems to be an emergent consensus around events like birth, merge, split, growth, expansion, contraction and death. Some authors propose additional events like continuation (i.e. no growth or expansion) and resurgence for communities that appear periodically (Rossetti and Cazabet, 2018). These key community events and their behavioral interpretations are summarized in Table 2.7 to provide a clearer comparison of event definitions used in previous studies. A summary of these events with informal definitions can be found in (Cazabet and Rossetti, 2019) as well as a formalism for lifecycle representation based on a directed graph where vertices are timed community observations and edges are continuation events bridging time gaps.

The consideration of events and transation needs to capture the changes occured. A module may merge or split into multiple modules at a later time step and it may survive if there exists a similar module in a future time step. Nevertheless, if there is no similar module, the module will dissolve.

Most of the work on dynamic network do not consider the reason why a community or an individual experiences a specific event or transition. For example, Nguyen et al. observed the dyamic network as a collection of simple events where a simple event can be one of newNode, removeNode, newEdge, removeEdge. NewNode is defined by a new vertex that appears in the network at the current time step, and maybe at the later time step the vertex is removed, it is called removeNode. QCA is only tested on real world network such as ENRON email network, arXiv e-print citation network and Facebook network. The

bright applicability of QCA is via a realistic application on routing strategies in Mobile Ad Hoc Networks (MANETs) in wireless network field (Nguyen et al., 2011). Most of the work on dynamic networks do not consider the reason why a community or an individual experiences a specific event or transition.

To provide a clearer understanding of these behaviors and their representation in existing studies, Table 2.7 summarizes key findings and event types identified in previous research. The table effectively summarises existing techniques in dynamic community detection and also highlights several critical research gaps. Most reviewed techniques rely heavily on local or pairwise similarity measures and lack a mechanism to integrate multi-level neighbour connectivity when computing module attraction. Consequently, these approaches tend to perform well in static or low-dynamic environments but struggle to maintain stability when communities evolve through frequent merge-split events. Moreover, few techniques explicitly justify the theoretical basis for their chosen connectivity depth, often treating it as a fixed parameter rather than a structural property of the network.

Table 2.7 Summary of performance evaluation and behaviour in dynamic network from previous studies

Author	Year	Name of DCD	Performance Evaluation	Name of event used as an effect dynamic network	Synthetic network	Real-world network
Lin et al.	2009	FacetNet	1) Qs 2) Mutual information	1) Inserting nodes 2) Removing nodes	1) White et al. 2005, static network 2) Newman and Girvan. 2004, dynamic network	1) NEC Blog 2) DBLP Co-authorship
Nguyen et al.	2011	QCA	1) Q 2) NMI	1) newNode 2) removeNode 3) newEdge 4) removeEdge	-none-	1) ENRON network 2) arXiv network 3) Facebook network. 4) MANET
Folino & Pizzuti	2014	DYNMOGA	1) NMI 2) Error rate	1) Birth and death 2) Expansion and contraction 3) Intermittent communities 4) Merging and splitting	1) Extended LFR network 2) SYN-FIX 3) SYN-VAR	1) Cell Phone Calls 2) Enron mail
Agarwal et al.	2018	DyPerm	1) NMI 2) ARI	1) newNode 2) removeNode 3) newEdge 4) removeEdge	Extended LFR network	1) High school contact networks 2011 2) High school contact networks 2012 3) Primary school contact networks 4) Contact workplace network 5) Cumulative coauthorship network 6) Noncumulative coauthorship network

Author	Year	Name of DCD	Performance Evaluation	Event used as an effect dynamic network	Synthetic network	Real-world network
Su et al.	2020	IncNSA	1) Q 2) NMI	1) Insert new community 2) Removed nodes 3) Newborn nodes 4) Remove the disappeared nodes	Extended LFR network 1) Birth and death 2) Expansion and contraction 3) Merging and Splitting	1) Phone call 2) AS-Oregon 3) HEP-TH 4) AS-Internet 5) Enron Email
Sun et al.	2020	DCDID	1) NMI 2) ARI	1) Add nodes 2) Delete nodes 3) Add edges 4) Delete edges	Extended LFR network 1) Nodes switch 2) Birth and death 3) Growth and contraction 4) Merger and Split	1) High school contact networks 2011 2) High school contact networks 2012 3) Primary school contact networks 4) Contact workplace network 5) Cumulative coauthorship network 6) Noncumulative coauthorship network
Sun et al.	2022	DCDME	1) NMI 2) ARI 3) Q	1) Adding nodes in a community/ between communities 2) Deleting nodes in a community/ between communities 3) Adding edges in a community/ between communities 4) Deleting edges in a community/ between communities	Extended LFR network 1) Nodes switch 2) Expansion and Contraction 3) Birth and death 4) Merger and Split 5) Mixing Parameter 6) Community density	1) Primary school contact networks 2) Contact workplace network 3) Cell phone call network 4) Enron email network 5) Cumulative coauthorship network 6) Noncumulative coauthorship network

2.6 Performance Evaluation for Dynamic Community Detection

In literature, evaluating community detection is a critical aspect of understanding their effectiveness and stability in uncovering meaningful structures within networks. This evaluation process involves comparing detected communities against predefined benchmarks or "ground truth" to assess accuracy, examining the stability of the detected communities under varying conditions, and validating the overall results to ensure they reflect the true characteristics of the network. By systematically evaluating these factors, the strengths and limitations of different dynamic community detection are referred to in Table 2.7. Summarization of the evaluation criteria that were applied later in the experiment is listed in Table 2.8. The goal of our analysis was to evaluate the effectiveness and stability of community structure detection using different methodologies across two metrics: NMI, and ARI.

Table 2.8 Summary of evaluation criteria settings for stability issue

Category Criteria	Suitable Method	Description	Purpose
Network Dynamics	Temporal Evaluation Metrics	Use a few metrics such as NMI and ARI over time to evaluate community detection adaptation to network changes.	To assess the stability and accuracy of community detection as the network evolves
Real-world Validation	Cross-Validation on Real-world Datasets	Validate using benchmarks such as Ground Truth Comparison, F1-score, Precision-Recall, and ROC curves.	To confirm that the performance is reliable and applicable in practical or real-world scenarios.

2.6.1 Stability Metric

The stability of a community detection refers to its capacity to deliver consistent results when applied to similar or slightly altered versions of the same network. This concept emphasizes how effectively the network's structure aligns with the detected community structure. To assess stability, NMI, and ARI serve as external evaluation metrics by comparing the technique's output to a ground truth, while modularity functions as an internal evaluation metric by assessing the community structure based on the network's structural properties.

In the context of this thesis, stability metrics are crucial for validating the stability of the proposed dynamic community detection technique. It helps to determine whether the technique can maintain consistent community partitions despite temporal or structural variations in the network.

2.6.1.1 Normal Mutual Index

NMI is a normal mutual index proposed by Danon et al. (2005) and is used as a metric to evaluate the effectiveness and community structure correctness of different methodologies. NMI has been proven as reliable and is currently used in testing community detection (Lancichinetti and Fortunato, 2009). A well-known information theory function of NMI is represented in Equation (2.9) as follows:

$$NMI = \frac{2I(U;V)}{H(U)+H(V)} \tag{2.9}$$

where U and V are the partitions of communities, I(U;V) denotes the mutual information of random variables U and V, and H(U) represents the entropy of U. NMI represents the degree of dependence between the partitions. When the ground truth is known, the NMI can be used to compare the partitions given by proposed techniques to the ground truth and non-overlapping community identification. NMI computations calculate the similarity (mutual information) between network groups, which indicates a community detection technique's stability and homogeneity. The value of NMI ranges from 0 to 1, where 0 represents that the detected communities are completely independent of the real communities, and 1 denotes a perfect match with the ground truth. A higher NMI score indicates a better alignment between the detected communities and the actual structure of the network.

In this study, NMI was used to evaluate the correspondence between the detected communities and the known or reference community structures within dynamic networks. This ensures that the proposed technique not only adapts to network evolution but also preserves structural coherence across time steps.

2.6.1.2 Adjusted Rand Index

Meanwhile, another metrics called adjusted rand index, ARI was introduced by Vinh et al. (2010) to evaluate the similarity between two communities. The definition of ARI is explained in Equation (2.10) as follows:

$$\mathrm{ARI} = \frac{\mathrm{RI} - \mathrm{RI}_{\mathrm{expected}}}{\mathrm{RI}_{\mathrm{max}} + \mathrm{RI}_{\mathrm{expected}}} \tag{2.10}$$

where R is the random and I is the index. RI denotes the similarity of two partitions, which includes all pairs of samples. Next, it calculates the number of pairs assigned to the same or

different partitions in the predicted and true partitions (Hubert and Arabie, 1985). It quantifies the accuracy of community detection results, with a higher ARI indicating better performance. More details can be found in (Vinh, Epps and Bailey, 2010).

Within the scope of this thesis, ARI was employed to measure how accurately the proposed technique reproduces true community assignments over time. It provides an additional layer of validation for the stability of the detected communities, especially in comparison to other dynamic community detection.

2.6.2 Ground Truth Network

A ground truth network refers to a network where the true structure and community assignments of vertices are known in advance. In many studies, especially those involving network analysis and community detection, ground truth networks serve as a benchmark for evaluating the performance of various techniques. The communities or modules within these networks are predefined based on prior knowledge, allowing researchers to assess how well the technique can recover these known structures. This concept is critical in the field of network science because it provides a means of validation for theoretical models and computational methods. Ground truth networks are often constructed from real-world data where the relationships and groupings among entities are well understood, such as in social networks, biological networks, or technological networks.

In research, ground truth networks provide a basis for fair and reliable comparisons of community detection techniques. For instance, Lancichinetti et al. introduced benchmark networks with known community structures, which have since become a standard tool for testing and comparing the effectiveness of static community detection (Lancichinetti,

Fortunato and Radicchi, 2008). Another popular synthetic benchmark for dynamic networks is an extended Lancichinetti-Fortunato-Radicchi (LFR) proposed by Greene et al. for dynamic community detection (Greene, Doyle and Cunningham, 2010). The extended LFR network is an acceptional synthetic network because it features heterogeneity in both the degree distribution of vertices and the distribution of community sizes, which is common in dynamic networks. It was proven by previous researchers in Section 2.5 about behavioural and role analysis.

To investigate the real-world network of community detection, researchers frequently consult publications. These publications provide valuable insights into the practical uses of community detection across various domains, especially in social networks. It is commonly used to explore interactions within email networks, among students, or between individuals in a company. Additionally, community detection is applied to analyze collaborations between authors in publication databases.

Pranckutė explored the use of Web of Science (WoS) and Scopus, a prominent bibliographic database widely used by researchers, academics, and institutions to access and investigate scholarly literature (Pranckutė, 2021). Scopus offers a comprehensive collection of peer-reviewed articles, conference papers, and other scholarly publications, making it essential for academic research. Created by Elsevier, Scopus is one of the largest databases of abstracts and citations (Singh et al., 2021). It covers a wide range of subjects, including natural sciences, social sciences, engineering, and medicine. Scopus allows researchers to search and retrieve relevant articles, track citations, and analyze publication trends. It provides extensive coverage of scientific research and offers insights into the academic impact and influence of various publications.

The comprehensive review about the network dataset used for dynamic community detection is explained in detail in Section 3.2.3. By using these networks, researchers can quantify the accuracy of their methods through metrics such as NMI, and ARI which compare the detected communities against the ground truth. Such evaluations are essential for advancing the field, as they provide insights into the strengths and limitations of different techniques under various conditions, ultimately guiding the development of more robust and accurate community detection techniques.

2.7 Performance Validation using Statistical Approach

To validate the results of the performance evaluation, statistical approaches such as the Friedman test is employed to assess and demonstrate the stability of the proposed dynamic community detection. In performance comparison studies, several statistical approaches such as the paired t-test, Wilcoxon signed-rank test, ANOVA, or Bonferroni correction can be used. However, these approaches often assume normality and homogeneity of variances, which are not always satisfied in network data or non-linear community detection outcomes.

Therefore, the Friedman test is selected because they are non-parametric, distribution-free, and well-suited for ranking multiple algorithms across multiple datasets or time steps. These tests provide a robust means to compare the relative performance of several techniques without assuming that the underlying data follow a normal distribution. This makes them particularly appropriate for evaluating dynamic network analyses, where the data are often irregular, sparse, or non-stationary.

2.7.1 Friedman Test

In this thesis, the statistical approach used was the Friedman test. The Friedman test is a non-parametric test that is applied to assess whether there are significant differences among three or more measurements from the same group on a skewed variable of interest. The Friedman test is sometimes called the Non-Parametric Repeated Measures ANOVA, Non-Parametric Friedman Test, or the Friedman Rank Sum Test. The Friedman statistic equation is shown in Equation (2.11).

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{2.11}$$

where, N is the number of observations or time steps, k is the number of groups or techniques, R_j is the sum of ranks of each technique j across all time steps, and χ_F^2 is the Friedman test statistics that follows a chi-squared distribution with $k-1$ degree of freedom.

The variable of interest should be continuous and exhibits a similar spread across the groups. Additionally, each group must contain sufficient data (more than 5 values). A hypothesis test is conducted at a 95% confidence level ($\alpha = 0.05$), where the null hypothesis states that all methods are equivalent, meaning their average ranks are equal.

For example, when comparing four techniques, the technique with the highest score on a given dataset is assigned 4 points, the second-highest receives 3 points, and so forth. This ranking process is applied across all datasets. Subsequently, the average rank for each technique is calculated across all datasets, providing a basis for comparing the overall performance of each technique in terms of quality metrics across different datasets.

The Friedman test was chosen because it effectively identifies whether there is a statistically significant difference among multiple techniques under comparison without

relying on data normality assumptions. This is advantageous when evaluating various dynamic community detection techniques across evolving network structures. This measure is useful for interpreting the practical significance of a statistical test. For validation of result in each phase, please refer to Chapter 5.

2.8 Research Gap

Despite the progress in community detection, maintaining stability and continuity in dynamic networks remains a major challenge. Existing dynamic community detection techniques such as modularity-based, label propagation, and spectral methods, perform well in static conditions but often produce inconsistent results when networks evolve. The inability to maintain stable community structures across time steps reduces the reliability of these techniques in representing real-world dynamic behaviors.

A key limitation arises from the use of traditional similarity measures, such as Jaccard, Salton, and Common Neighbors. These measures rely heavily on static topological relationships and fail to adapt to temporal variations caused by vertex and edge changes. Consequently, communities formed at different time steps may not remain coherent or comparable. In addition, current approaches to module attraction often consider only limited connectivity levels, neglecting the influence of neighboring modules and higher-order interactions. This simplification weakens the representation of evolving community relationships, leading to instability during community merging or splitting events.

Furthermore, several nature phenomena-inspired models including those based on the Matthew effect, fish school effect, and information dynamics have contributed valuable insights into adaptive community formation. However, these techniques still lack

mechanisms to balance attraction and separation forces effectively over time. As a result, they struggle to maintain consistent tracking of communities during rapid network evolution. Hence, a more biologically grounded and mathematically structured approach is required to achieve stability.

Finally, many studies have relied on performance metrics such as modularity, NMI, and ARI without statistical validation. The absence of non-parametric tests like Friedman test limits the reliability of comparative analyses. Therefore, there is a need for a stability-oriented and statistically validated proposed technique that integrates similarity measure and module attraction, ensuring stability detection of community structures in evolving networks.

2.9 Summary

Chapter 2 provides an in-depth exploration of the relevant literature on static community detection and dynamic community detection measures. It begins by introducing the two categories of dynamic community detection: independent and dependent. The dependent dynamic community detection can be divided into two, evolutionary and incremental approaches. However, this study highlights the incremental approach. While numerous techniques have been introduced, many still struggle with issues of stability when dealing with evolving network structures. These limitations provide the foundation for identifying a clear research gap.

Then, exploring the similarity measure, vertex attraction and module attraction techniques for the process of community detection. Next, the chapter examines the behaviour and role analysis of dynamic community detection from the previous study.

Additionally, it identifies the shortcomings of existing similarity measures to achieve a more comprehensive understanding of the quality of communities detected in complex networks. Another thing emphasized in this chapter is the evaluation metric for community detection. The chapter also addresses and outlines the research gap that this study aimed to fill.

Building upon these insights, the next chapter outlines the research methodology adopted in this study. It explains the systematic framework, network datasets, and experimental design that guided the development and validation of the proposed dynamic community detection technique inspired by natural phenomena.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This chapter focuses on the research methodology of this study, which aims to enhance the identification of communities within network structures over time. The research methodology is outlined in detail to guide the study. It begins with the research framework, which serves as a guide for the entire research pipeline. Then, the chapter discusses the network datasets and compares the existing techniques with the proposed technique.

3.2 Research Framework

There have been several work focusing on improving network analysis pipelines for community detection on dynamic networks. The existing work mostly focused on the incremental approach that falls into the global method and involves the use of similarity measure to enhance community detection. The stability issue in incremental approach is a critical challenge due to the need to maintain consistent and meaningful communities as the network evolves. This problem is compounded by the sensitivity of techniques to small changes, such as the addition or deletion of edges or vertices, which can lead to significant fluctuations in community structures. To address this, a stable dynamic community detection framework must focus on similarity analysis to ensure that detected communities remain cohesive over time. This consideration is crucial for developing techniques that can reliably capture the evolving nature of networks while providing meaningful insights.

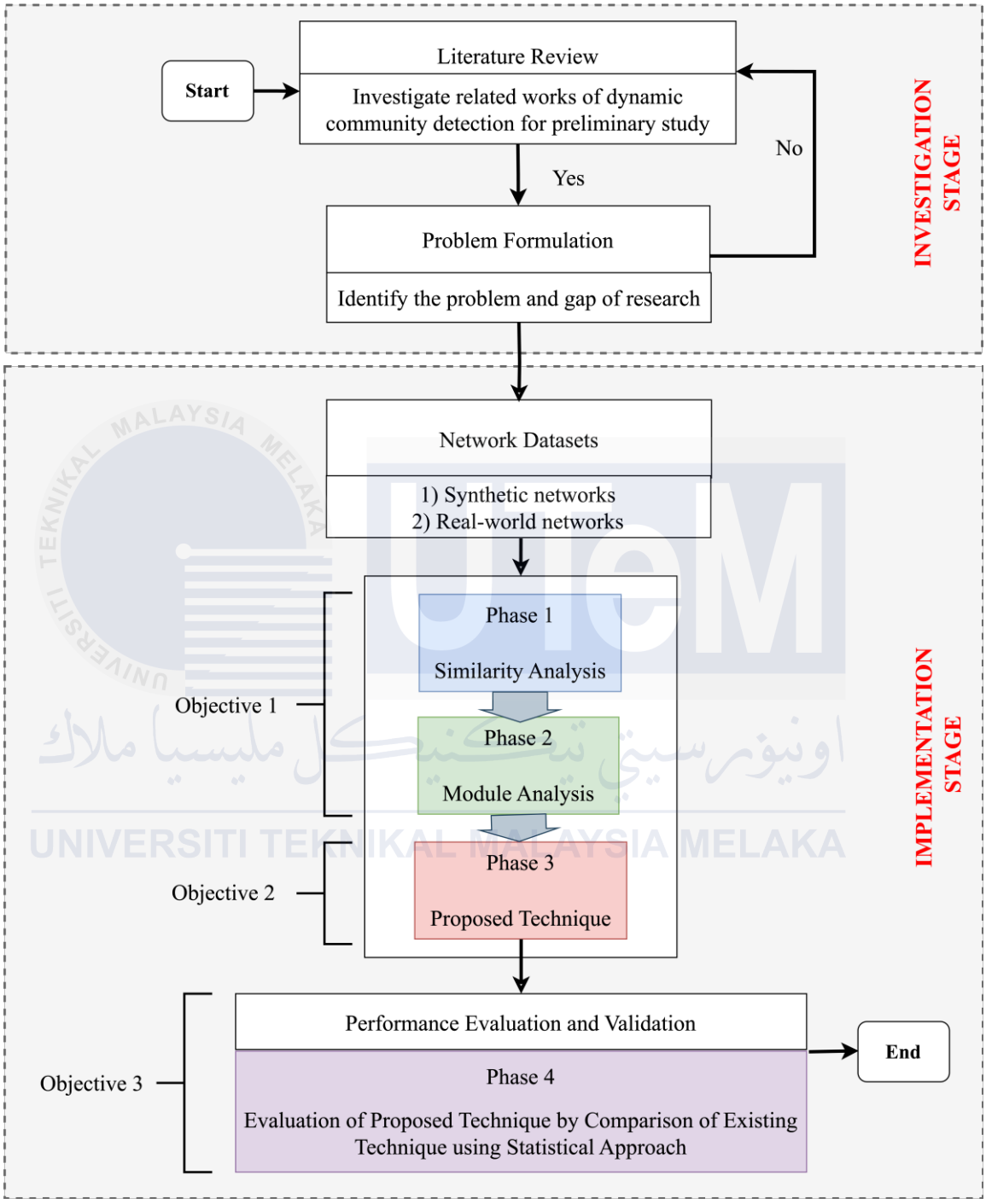


Figure 3.1 Research framework of dynamic community detection in this thesis

Similarly, this research adopted incremental approach as well as introducing enhancement of module attraction. Figure 3.1 shows the research framework of dynamic

community detection proposed in this thesis. The research outline shows a systematic approach to addressing the challenges in dynamic community detection. In this figure, two main stages of this research; i) investigation, and ii) implementation, are shown.

3.2.1 Investigation Stage

In the investigation stage, this study began with a comprehensive literature review aimed at investigating existing works related to dynamic community detection. This initial step is crucial for understanding the current state of research and identifying gaps that need further exploration. Therefore, the problem of the research came from the investigation of literature with related works for preliminary study in which the insights gained were from the Scopus and WOS databases. The evolution of coummunity detection for static and dynamic community detection for dynamic network was explored and the explanation is provided in Section 2.2 and 2.3.

3.2.2 Implementation Stage

Upon the completion of investigation stage, the implementation stage began. To ensure the model's robustness and applicability across various scenarios, both synthetic and real-world network datasets were utilized. This stage was divided into three core objectives, each addressed in a separate phase. Phase 1 focused on similarity analysis, aiming to develop techniques for effectively comparing network time steps. Phase 2 involved module analysis, exploring process to identify and track community structures within an evolving network. Building on the outcomes of the previous phases, Phase 3 introduced a novel

process that integrated similarity and module analysis, incorporating the bird flock effect to accurately detect dynamic communities. To evaluate the performance of the proposed technique, Phase 4 was dedicated to performance evaluation and validation, involving rigorous assessment through comparisons with existing techniques using statistical approaches. Through these structured phases, the research sought to advance the dynamic community detection techniques.

3.2.2.1 Phase 1: Similarity analysis

As discussed earlier in Section 2.4, three critical aspects of the community detection process were outlined. The first aspect, similarity analysis, serves as the foundational focus of this thesis. A major challenge in studying the evolution of communities is selecting an appropriate similarity measure. The choice of a suitable similarity function significantly impacts the performance of community detection techniques. The effectiveness of these techniques may vary depending on whether the similarity index emphasizes local or global properties of the network structure. Consequently, the degree of each vertex plays a crucial role in reflecting the network's structural characteristics. This section examines how different similarity functions influence the quality of community detection.

The first phase of the research is illustrated in Figure 3.2. The structured approach adopted in this research study, focuses on dynamic community detection in networks and emphasizes the importance of selecting an appropriate similarity measure, as outlined in Objective 1. The process was divided into three phases, with Phase 1 (Similarity Analysis) being central to fulfilling Objective 1, which aimed to identify and select the most suitable similarity measure that accurately captures the dynamic relationships within the network.

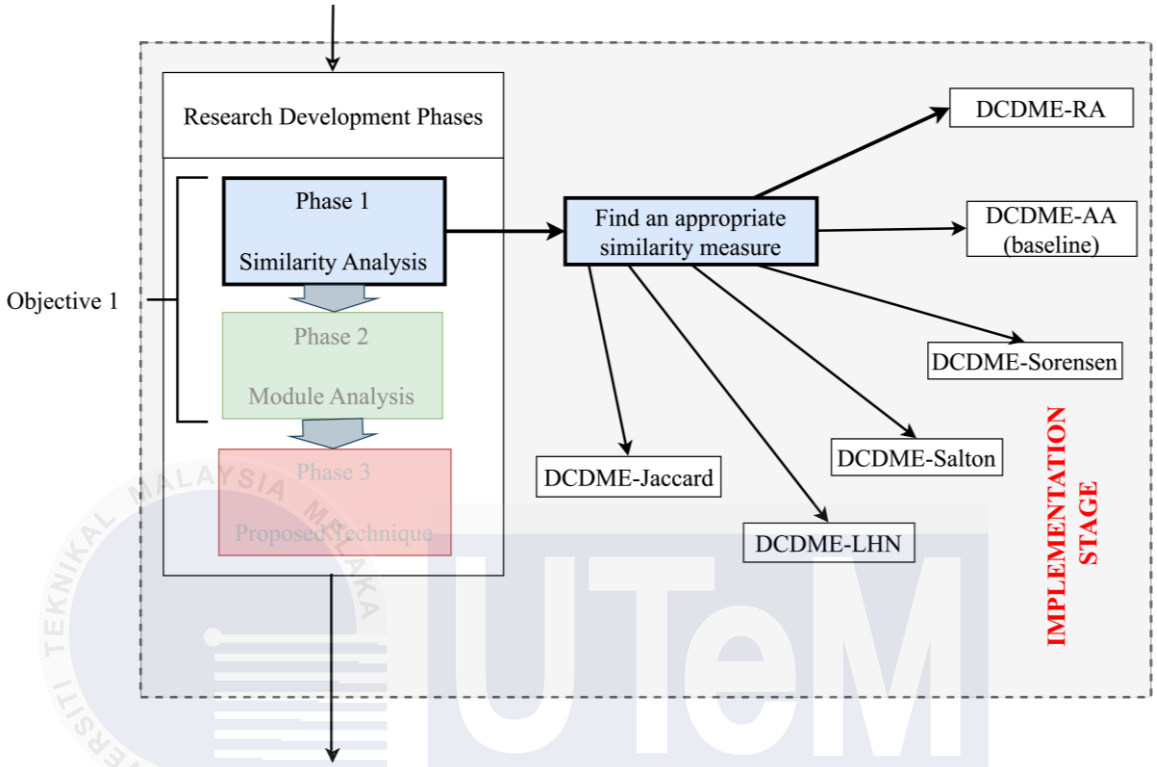


Figure 3.2 Phase 1: Similarity analysis by six types of similarity measure

Each similarity measure was applied to track community structures over multiple time steps with predefined events such as merger, split, expansion, and contraction. Based on a review of existing dynamic community detection techniques, the DCDME was selected as a baseline for this study. Within the scope of this research, six different similarity measures were evaluated: DCDME-RA, DCDME-AA, DCDME-Sorensen, DCDME-Jaccard, DCDME-Salton, and DCDME-LHN. The best-performing measure, identified as DCDME-RA, corresponded to the Resource Allocation (RA) index. The evaluation employed three quantitative metrics: Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and execution time. For each dataset, experiments were repeated ten times and were recorded to minimize random variation. The "appropriate" similarity

measure was defined as the one that achieved the highest average NMI and ARI values, indicating the most consistent community alignment with ground truth, while maintaining reasonable computational efficiency.

Later in Section 5.2.1, the selection of RA index is explained. This research effectively addresses Objective 1, ensuring that the chosen similarity measure not only enhances stability in community detection but also improves the precision and adaptability of the technique in dynamic environments. This selection was anticipated to improve community detection results, though it requires a significant amount of computational time.

The term mesoscopic structure refers to the intermediate-level organization within a network, lying between the local (individual vertices and edges) and global (overall network) scales. It often encompasses community structures, clusters, or modules that reveal meaningful groupings of vertices based on connectivity patterns. Nonetheless, identifying an appropriate technique for gaining deeper insights into the mesoscopic structure of a dynamic network remains a challenging task, given the expected quality of results. Recent efforts have explored various qualities of community structures associated with detection techniques, yet the question remains complex.

The results of this phase, presented later in Chapter 5, were statistically validated using Friedman test to confirm the significance of the performance differences among the similarity measures. The impact of each similarity measure is discussed in terms of performance evaluation. This selected similarity measure was subsequently applied in the next phase of the research.

3.2.2.2 Phase 2: Module analysis

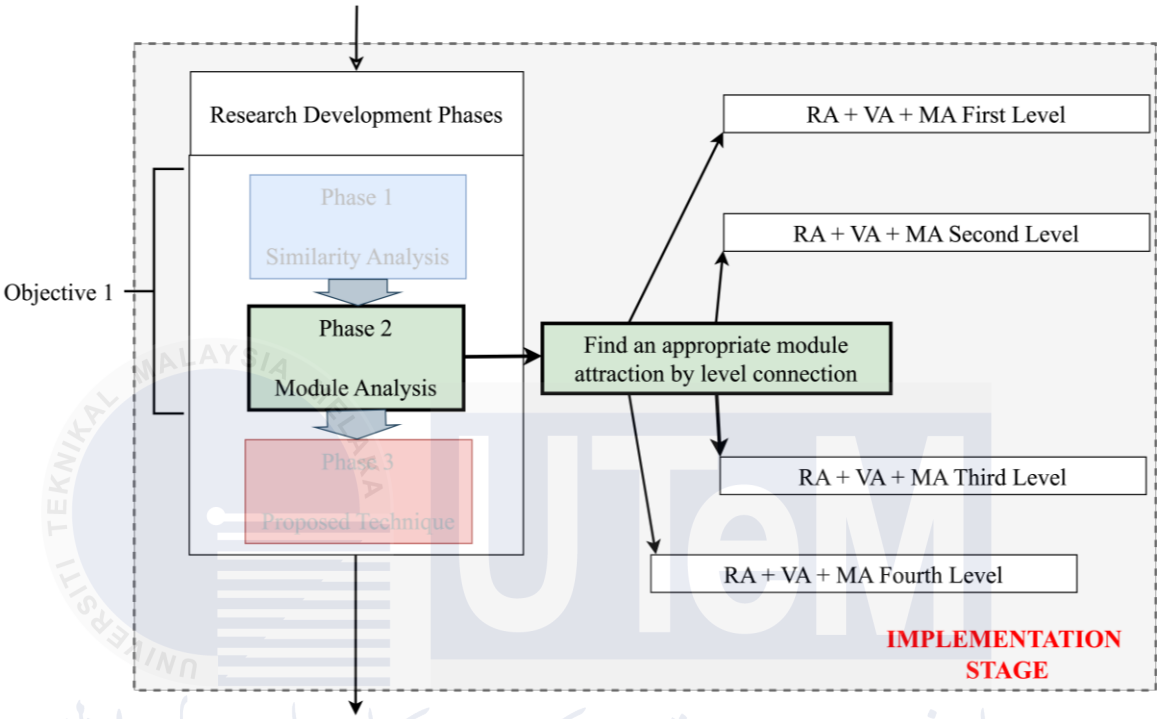


Figure 3.3 Phase 2: Module analysis based on the level of connectivity

The second phase of this research is shown in Figure 3.3. It continues the structured approach outlined in Figure 3.2, shifting the focus from similarity analysis to module analysis, which is essential for fulfilling Objective 1 of this research. The purpose of this phase was to gain a deeper understanding of the network's mesoscopic structure by identifying the most effective module attraction (MA).

In Phase 2: Module Analysis, this study examined how different levels of vertex attraction (VA), inherently linked to the RA index selected in Phase 1, influence the quality of community detection. Figure 3.4 illustrates the evaluation of four levels of vertex attraction, first level or 1 hop (direct), second level or 2 hops (neighbouring), third level or

3 hops (extended), and fourth level or 4 hops (distant) each representing varying degrees of connectivity within the network.

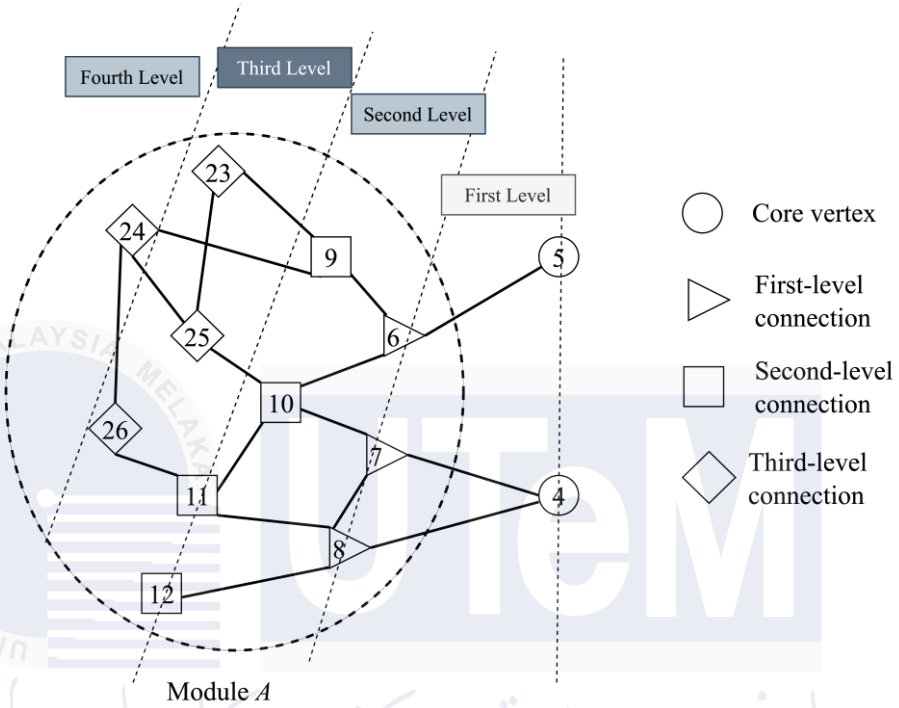


Figure 3.4 Illustration for level connection of module attraction

In network theory, community structure is understood to exist at the mesoscopic scale, which lies between local interactions (1–2 hops) and global network influence (4 or more hops). First- and second-level connections capture only the immediate and neighbouring vertices, which are often too local to reveal the broader cluster boundaries, especially in dynamic networks where vertices frequently change neighbours. In contrast, fourth-level and higher connections extend too far into the graph, mixing information from multiple communities and introducing structural noise that dilutes true module identity.

The third-level neighbourhood represents the smallest radius at which mesoscopic community cues emerge. At three hops, vertices begin to reflect the extended cohesion and

bridge patterns within their true community, while still remaining close enough to avoid contamination from unrelated or distant modules. This aligns with established principles in hierarchical network organisation and matches the typical radius at which community cores and peripheries interact. Thus, the third level offers the theoretically appropriate balance, broad enough to capture structural cohesion, yet constrained enough to prevent over-aggregation.

The evaluation metrics again included NMI and ARI, measured across multiple merge-split and birth-death scenarios. The "appropriate" level of module attraction was defined as the configuration that produced the highest stability and similarity with the ground truth across evolving time steps while minimizing abrupt community changes. Statistical validation using Friedman test, reported in Chapter 5, empirically supported the selection of the optimal connection level. The detailed results are discussed in Section 5.3, and the superior performance of the third level is a key finding in this research for Phase 3, where these findings are integrated into the development of the proposed technique.

3.2.2.3 Phase 3: Proposed Dynamic Community Detection Inspired by Bird Flock Effect

In the continuation of the research approach outlined in Figures 3.2 and 3.3, Figure 3.5 introduces the proposed technique in Phase 3, which draws inspiration from the bird flock effect. The third phase of this research represents the integration of the findings from Objective 1 (selection of the most effective similarity measure) and Objective 2 (determination of the optimal module attraction level). Together, these two components form the foundation of the proposed technique addressed.

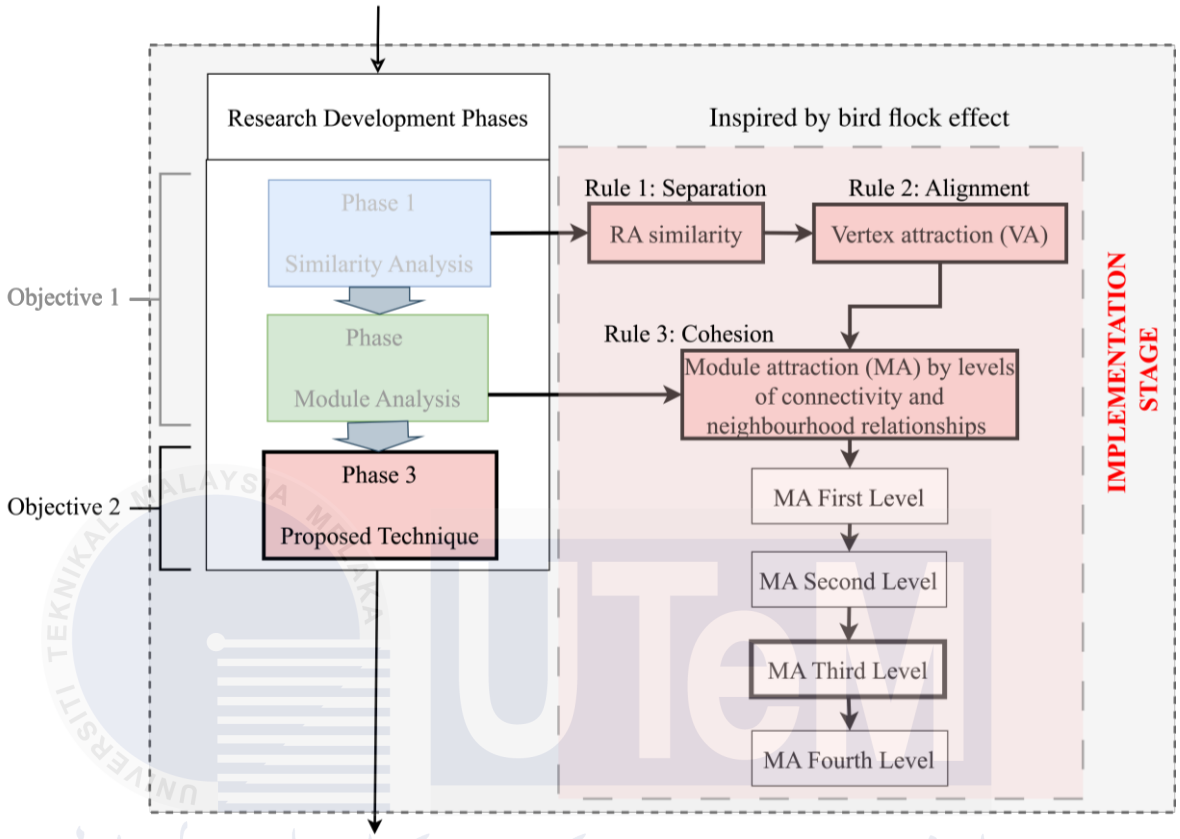


Figure 3.5 Phase 3: Proposed technique inspired by bird flock effect

Objective 3 serves as the overarching evaluation of the integrated technique. This effect, observed in nature, is governed by three fundamental rules from Craig Reynolds in 1987: Separation, Alignment, and Cohesion (Reynolds, 1987; Lijcklama à Nijeholt, 2020). These rules dictate the movement and formation of bird flocks and were adapted in this research to improve dynamic community detection in networks. This study reformulated the original rules and adapted the modification in network analysis.

First, rule 1: Separation was implemented using RA similarity, ensuring that vertices with lower connectivity maintain their distinctiveness while still contributing to the overall community structure. Second, rule 2: Alignment was achieved through vertex attraction,

aligning vertices with similar connectivity levels to ensure accurate and cohesive community formation. Finally, rule 3: Cohesion involved assessing different levels of connectivity, with the third level being identified as providing the best performance. This level strikes a balance between capturing significant connections and avoiding the noise from overly distant or overly close connections, which could otherwise compromise the accuracy of community detection. By integrating these rules, the proposed technique effectively leveraged the natural dynamics seen in bird flocks, enhancing the precision, stability, and adaptability of community detection in dynamic networks.

3.2.2.4 Phase 4: Evaluation of Proposed Technique by Comparison of Existing Techniques

In continuation of the research process outlined in the previous figures, Figure 3.6 presents the final phase, Phase 4: Performance Evaluation and Validation. This phase is crucial for fulfilling Objective 3, which entails evaluating the effectiveness of the proposed technique by comparing it with existing, well-known techniques using a statistical approach. The aim of this phase was to validate the performance of the developed method through rigorous testing and to ensure that improvements made in the earlier phases translate into measurable benefits.

In this study, the stability evaluation metric employed Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI). These metrics provide a comprehensive assessment of the technique's accuracy, stability, and overall effectiveness in detecting communities within dynamic networks.

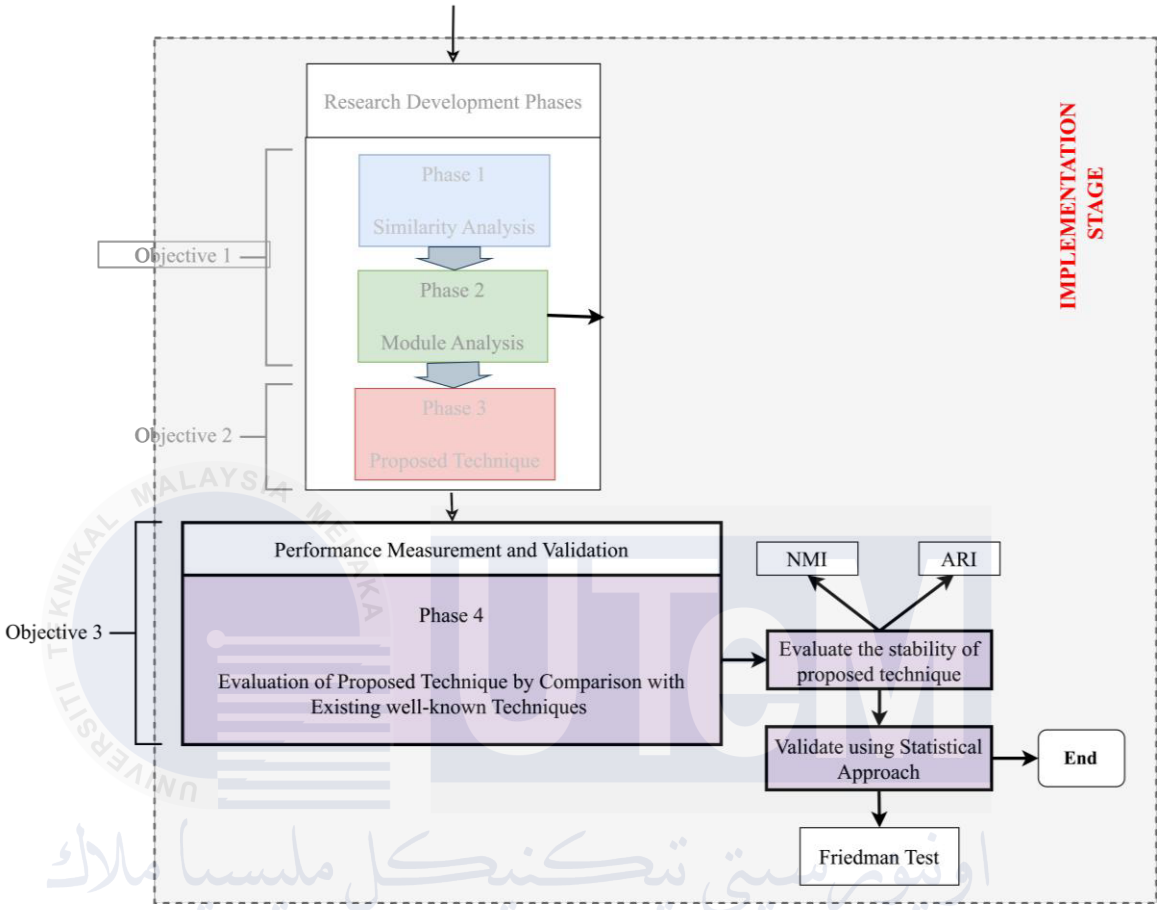


Figure 3.6 Phase 4: Evaluation and validation of the proposed technique

The performance of the proposed technique was compared against established techniques, with the results of these comparisons being essential for validating the technique's superiority or identifying areas for further refinement. By systematically applying these metrics, this study ensures that the proposed technique is not only theoretically sound but also practically effective in real-world scenarios, thereby achieving the study's final objective.

Performance validation employed statistical approaches, including the Friedman test to demonstrate the stability of dynamic community detection inspired by the bird flock effect. These non-parametric tests were selected due to their robustness in handling data that

may not meet the normality assumptions required by parametric tests. In dynamic network analysis, performance data often involve rankings, ordinal measures, or non-normally distributed results, making non-parametric statistical approach more suitable for accurate assessment. The Friedman test is effective for detecting differences across multiple techniques without assuming a normal distribution.

To perform the Friedman test, it is essential to begin by defining the null and alternative hypotheses, where the null hypothesis states that there is no significant difference among the groups and the alternative hypothesis posits that there is a significant difference between at least two of the groups. Following the establishment of these hypotheses, the test statistic and the corresponding p-value can be computed for the given dataset.

The test statistic was determined using the formula in Equation (2.11). Once it was established that a significant difference exists between the groups through the Friedman test, a post-hoc analysis was required to identify the specific pairs of groups that exhibit significant differences. By employing these non-parametric approaches, the study ensures that the evaluation results are both statistically sound and resilient to the inherent variability in dynamic network performance data. All techniques were executed on a device equipped with Python, running on an Intel Core i7-7700 CPU at 2.8GHz with 24GB of RAM.

3.2.3 Network Datasets

This study utilized secondary datasets, specifically dynamic network datasets that are publicly available on the internet and well-known as benchmark networks for researchers. The two types of network datasets were: synthetic networks and real-world networks. The inclusion of synthetic network serves a critical methodological purpose in which it enables controlled experimentation under known ground-truth community structures, which real-

world datasets rarely provide. Real-world networks provide valuable structural and behavioural patterns; however, they contain several inherent limitations that restrict their suitability for evaluating stability in dynamic community detection.

First, most of the real datasets do not provide ground-truth community structures across all time steps, making it impossible to objectively compute NMI and ARI for stability assessment. Second, the temporal evolution observed in real-world networks occurs naturally and uncontrollably, meaning that the number, scale, and type of events (e.g., merger, split, birth, death) cannot be systematically varied. This prevents a fair comparison between techniques when analysing stability under identical dynamics. Third, real-world network datasets often exhibit irregular sparsity, missing edges, and noisy interactions, which makes them unsuitable for isolating the effect of specific behaviours on algorithm performance.

To overcome these limitations, this study incorporated synthetic dynamic networks generated using the extended Lancichinetti-Fortunato-Radicchi (LFR) benchmark, which is widely recognised for producing realistic community structures with controllable parameters. Synthetic data allow the creation of dynamic scenarios where the rate, intensity, and type of structural changes can be precisely specified. This controlled environment is essential for evaluating the temporal stability of DCD techniques, particularly when the objective is to isolate the contribution of similarity measures, module attraction, and incremental updating. The synthetic datasets complement the real networks by enabling a systematic, statistically valid assessment of algorithmic behaviour under predefined dynamic conditions that do not exist in real-world data.

3.2.3.1 Data description for synthetic networks

One popular synthetic network that can be generated using computer is the LFR network. The LFR was provided by Lancichinetti and Fortunato which includes ground-truth communities known as true communities (Lancichinetti, Fortunato and Radicchi, 2008). In 2010, Greene et al. made improvements to the LFR model, adapting it for dynamic networks. This enhanced version is known as the extended LFR (Greene, Doyle and Cunningham, 2010). The extended LFR network is an exceptional synthetic network as it features heterogeneity in both the degree distribution of vertices and the distribution of community sizes, which are common in dynamic networks.

This extended LFR network allows the generation of networks with specific parameters providing a controlled environment for testing. In this benchmark, several key parameters define network structure and evolution: the mixing parameter, which determines the proportion of inter-community edges; the community density, representing the average number of edges per vertex within a community; and the vertex switching probability, which controls how frequently vertices change their community membership across time steps. These parameters are important for simulating realistic dynamic behaviors such as community expansion, contraction, merging, splitting, birth, and death as detailed in Table 3.1.

Table 3.1 Definition of parameters of the extended LFR

Symbol	Definition
n	Number of vertices in each time step
S	Number of time steps in the dynamic network
μ	Mixing parameter
k	Average degree of the dynamic network
p	Probability of vertex switching between two adjacent time steps
e	Number of community expands in every time step
c	Number of community contracts in every time step
b	Number of community births in every time step
d	Number of community deaths in every time step
m	Number of community mergers in every time step
s	Number of community splits in every time step

The selected values for μ, k, and p were derived from empirical studies and benchmark configurations commonly used in dynamic network research (Lancichinetti & Fortunato, 2009; Rossetti & Cazabet, 2018). The specific parameter ranges were designed to represent low, moderate, and high levels of network complexity and temporal change. Table 3.2 summarizes the evaluation criteria, parameter settings, and purposes of each experiment conducted in this study. This thesis selected three values to cover low, medium, and high dynamic behaviours, which are the standards in LFR-based benchmark studies and provide sufficient variation to observe trends without creating unnecessary experimental complexity.

Table 3.2 Evaluation criteria, setting and purpose of experiment in synthetic networks

Evaluation criteria	**Parameter Setting**	**Purposes**
Effect of Merger and Split	*m*: 5, 20, 40 *s*: 5, 20, 40	To determine the impact of network mergers and splits on community coherence and stability of performance.
Effect of Expansion and Contraction	*e*: 5, 20, 40 *c*: 5, 20, 40	To evaluate how changes in network expansion and contraction impact community detection and structure accuracy.
Effect of Birth and Death	*b*: 2, 8, 16 *d*: 2, 8	To assess the influence of vertex birth and death events on the stability of detected communities.
Effect of Mixing Parameter	μ = 0.2, 0.4	To analyze the effect of varying mixing parameters on the overlap between communities and the robustness of community detection.
Effect of Community Density	*k*: 5, 15, 25	To explore how different community densities affect the precision of community detection.
Effect of Vertex Switching	*p*= 0.1, 0.4, 0.8	To investigate how vertex switching impacts the consistency and sensitivity of community detection under varying network structures.

The number of vertices was fixed at n = 1000 to represent a medium-scale network that balances structural complexity with computational feasibility. The minimum community size S = 20 aligns with typical sizes observed in social and collaboration networks, ensuring the presence of small-to-medium communities while preventing overly fragmented structures. Degree-related parameters, including the average degree (k = 5, 15, 25) and maximum degree ($k$max = 20), were chosen to emulate realistic heavy-tailed degree distributions and to avoid generating unrealistic hubs.

Varying the mixing parameter μ = 0.1 and 0.4 enabled evaluation under both well-separated (low μ) and highly mixed or noisy (high μ) community structures, conditions frequently encountered in temporal social networks. Community detection became harder as mixing parameter increased. Dynamic event parameters were varied to reflect a wide spectrum of real-world network transitions. Extended LFR benchmark could not generate time step when community births and deaths both reached 16, therefore no $d = 16$ variety. The summary for synthetic networks is shown in Figure 3.3.

3.2.3.2 Data description for real-world networks

For empirical networks, also referred to as real-world networks, six real-world datasets were employed, with detailed summary provided in Table 3.3. Each dataset exhibited distinct structural and temporal characteristics that justify its inclusion in this study. These differences are essential for evaluating the performance, adaptability, and stability of the proposed DCDBFE technique under diverse real-world conditions. A description of each social network dataset is provided below:

1. **Primary School dynamic networks**: This dataset is characterized by highly structured and periodic interactions between students and teachers. A student-teacher social network and the data set contain nine-time steps in a network of contacts between children and teachers. Each student and teacher represents a vertex, and the contact between them represents an edge. Communities correspond to classroom groups, which remain relatively stable over time. The controlled environment and clearly defined group boundaries make PS suitable for analyzing the stability of community detection under low-noise, well-separated community structures.

2. **Workplace Contact dynamic networks**: This dataset contains moderately stable but cross-department interactions, with five real departments forming the ground-truth communities. Compared to PS, WC is more heterogeneous, reflecting professional interactions, movement between office spaces, and informal contacts. A social network of contacts between people in a French office building. The time series has five departments as real communities and recorded contact between people at intervals is 20 seconds and eight-time steps. The community structure is clear but subject to non-periodic and inter-department link fluctuations, enabling assessment of stability in environments with moderate structural variation.
3. **High School 2011 dynamic networks**: These networks represent adolescent interaction patterns, which are more dynamic and mixed than school or workplace networks. A social network of connections was formed between students by three classes of a high school in Marseille, France in December 2011. HS2011 has seven time steps and each of it has three communities. In HSD11 dynamic network, one vertex represents a student, and one edge indicates that there is a connection between the students. HS2011 has smaller class groups (three communities) and fewer time steps, making it suitable for evaluating detection consistency under moderate dynamics.
4. **High School 2012 dynamic networks**: Similar to HS2011, a social network of connections was formed between students by five classes of a high school in Marseille, France. The time series was in November 2012. The other information is consistent with HSD11. HS2012, with five communities and more interaction diversity, exhibits higher student mobility and greater variation in community boundaries.

5. **Cumulative Coauthorship Network (CC)**: This dataset represents a growing collaboration network where vertices and edges accumulate over time. Communities tend to expand, merge, or evolve slowly as researchers collaborate on more publications. The version used in this study was curated and modified based on the reference from Chakraborty, T. et al., 2014. In this dynamic network, each vertex represents an author, while edges denote coauthorship relationships. The community structures gradually densify and evolve but rarely disappear. It tests the technique's ability to maintain long-term stability across evolving but persistent communities.
6. **Noncumulative Coauthorship Network (NCC)**: Similar to the CC network, this dataset also represents a collaboration network. However, unlike the CC network, the NCC dynamic network does not accumulate changes. In this case, even if two authors coauthor multiple papers, only a single link exists between them, regardless of how many collaborations occur. This produces a more volatile network with temporal fluctuations influenced by research cycles. Community structures appear, dissolve, and re-form over time.

The inclusion of these six datasets ensures that the evaluation of the proposed DCDBFE technique is comprehensive and reflective of real-world dynamic behaviour. Each dataset represents a distinct structural and temporal profile, ranging from highly stable, well-defined communities to rapidly fluctuating, weakly structured interaction patterns. This diversity enables rigorous assessment of the technique's stability across multiple domains. Consequently, the experimental findings can be generalised with greater confidence, demonstrating that DCDBFE can handle a broad spectrum of realistic dynamic network scenarios.

Table 3.3 The real-world network features in this study

No	Real Networks	Features				
		Vertices	Edges	Average number of degrees	Average clustering coefficient	Number of Time Steps
1	PS	239	6146	50.8	0.52	9
2	WC	88	537	11.4	0.38	8
3	HS2011	123	1271	20	0.51	7
4	HS2012	175	1629	18	0.43	8
5	CC	107 180	376 567	4.3	0.49	17
6	NCC	107 166	376 567	4.3	0.49	17

3.3 Comparison Between Existing and Proposed Technique

This section emphasizes the comparison between two dynamic community detection techniques: the existing Dynamic Community Detection based on Matthew Effect (DCDME-baseline) and the proposed Dynamic Community Detection based on Bird Flock Effect (DCDBFE). The comparison between the existing Dynamic Community Detection based on the Matthew Effect (DCDME-baseline) is shown in Figure 3.7 and the proposed Dynamic Community Detection based on Bird Flock Effect (DCDBFE) that highlights key methodological differences which contribute to enhancing the stability of community detection. Both approaches start by constructing the network and calculating vertex attraction to gauge the influence of individual vertices. However, the DCDME-baseline uses AA similarity (Adamic-Adar) as its similarity measure, guided by principles of the Matthew Effect, which emphasizes how high-degree vertices increasingly attract connections.

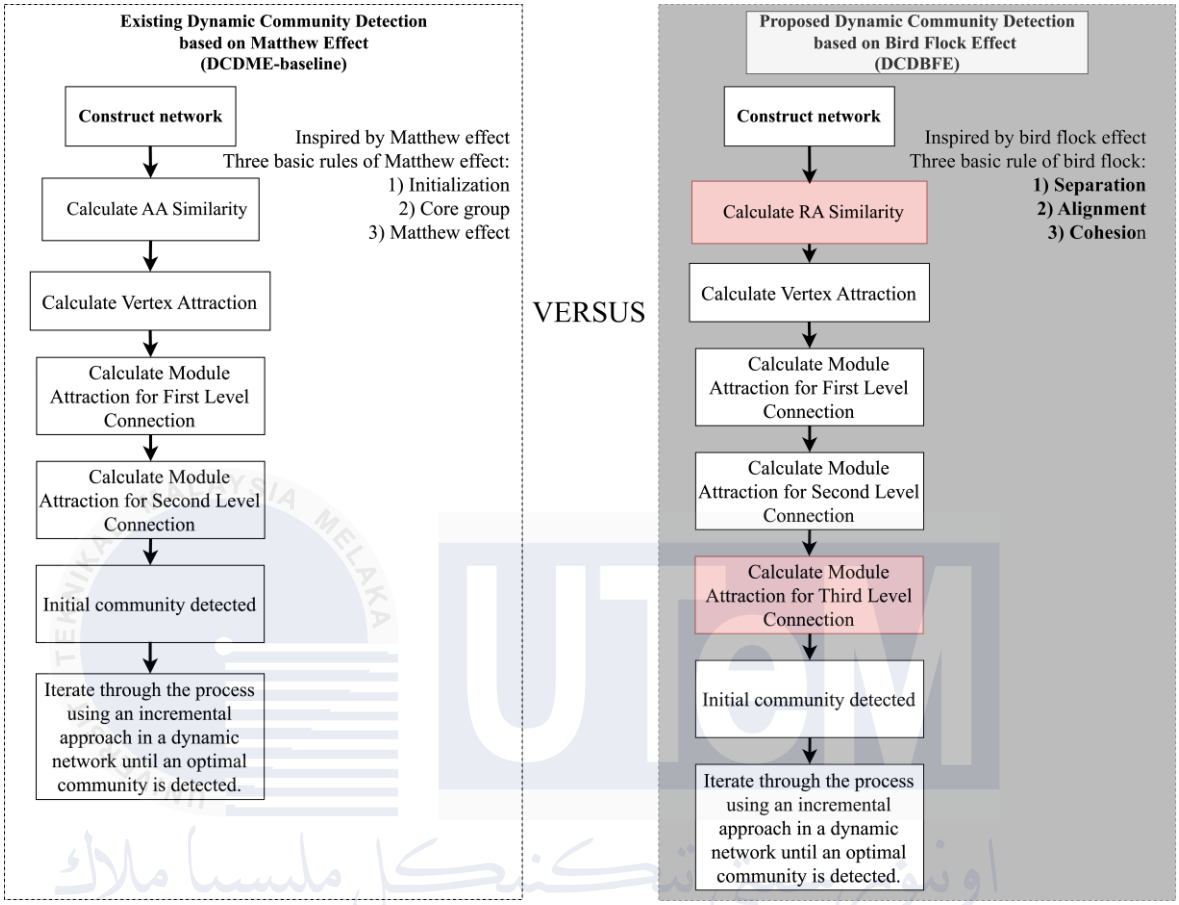


Figure 3.7 Comparison between existing and proposed technique

This method applies two levels of module attraction first and second level connections representing two stages of community aggregation, after which an initial community structure is detected. The DCDME-baseline then iterates using an incremental approach, adjusting the community structure dynamically until an optimal configuration is achieved.

In contrast, the proposed DCDBFE technique introduces significant improvements, particularly by incorporating RA similarity (Resource Allocation) and an additional third level of module attraction, inspired by the bird flock effect. The RA similarity measure was chosen for its capacity to capture relationships in a more adaptive and decentralized way,

aligning with the three basic bird flocking principles: separation, alignment, and cohesion. This model enables entities in the network to interact based on localized rules, which collectively result in more stable and coherent community structures. The third level of module attraction in DCDBFE provides a finer layer of community detection, capturing subtle connections within the network that might be missed by a two-layer approach. Module attraction is computed up to the third level of connections to capture meaningful meso-level structural cohesion. Levels beyond the third introduce long-range noise, while fewer levels fail to represent sufficient community structure. This ensures that the model balances local accuracy with global context.

By iterating through three levels of connection and employing the bird flock-inspired behavior, DCDBFE demonstrates enhanced stability and accuracy in identifying dynamic communities, particularly in networks where interactions evolve continuously. This additional layer in the DCDBFE approach not only refines the granularity of community boundaries but also improves adaptability in fluctuating network environments, establishing it as a stable alternative to the DCDME-baseline. Therefore, further empirical evaluation is necessary to substantiate these claims and determine the effectiveness in various real-world scenarios.

3.4 Summary

This section emphasizes the significance of a preliminary investigation stage and offers an in-depth explanation of the implementation stages used for a thorough analysis, laying a strong foundation for future research. The purpose of this chapter is to provide a

comprehensive overview of the approach guiding the subsequent stages of this research project, detailing the structure and rationale behind the selected techniques.

Chapter 4 introduces the implemented techniques, building on the processes previously defined. Therefore, the proposed dynamic community detection in this research is described as an enhanced dynamic community detection, based on an incremental approach inspired by the flocking behavior of birds, named DCDBFE. In the chapter, the conceptual framework of DCDBFE is translated into biologically inspired designed to improve the stability of community structures in dynamic networks.

CHAPTER 4

PROPOSED DYNAMIC COMMUNITY DETECTION

4.1 Introduction

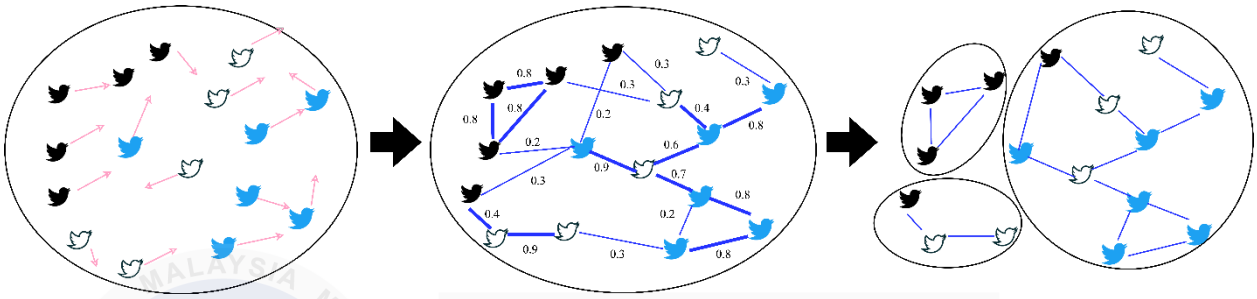
This chapter presents the enhanced dynamic community detection in this study. The technique was designed based on the inspiration from bird flock effect to identify communities over time based on an incremental approach. The core idea behind this technique is to apply natural phenomena that replicate the formation process of bird flocks and enable the analysis of their temporal evolution. The proposed technique, named Dynamic Community Detection Based on the Bird Flock Effect (DCDBFE), is specifically tailored to uncover community structures in dynamic networks. Then, the chapter discusses the pseudo code used throughout this chapter.

4.2 Basic Idea

Before delving into the proposed technique, it is essential to introduce some definitions that will be used in this chapter. These foundational concepts will provide the necessary background to understand the detail of the dynamic community detection that is inspired from the bird flock effect model.

In physical world, people are attracted by activities according to their interests, a phenomenon similar to the bird flock effect in the sky. The concept behind the formation process of bird flock effect where the technique is designed, is based on three basic rules from Craig Reynolds in 1987 (Reynolds, 1987; Lijcklama à Nijeholt, 2020). Figure 4.1 illustrates the formation process of bird flock effect model through three fundamental rules:

Separation, Alignment, and Cohesion. Each rule corresponds to a phase in the dynamic behaviour of entities within a network, and together, they help in understanding how communities form and evolve.



(a) Separation Phase (b) Alignment Phase (c) Cohesion Phase

Figure 4.1 The formation process of bird flock effect model

In the Separation phase, depicted in Figure 4.1(a), each bird (or network entity) maintains a certain distance from its neighbors to prevent overcrowding. This is visually represented by the arrows pointing away from one another, ensuring that the entities do not cluster too closely at this initial stage. The concept here is similar to individuals in a social network who begin by exploring their surroundings independently, without forming any close ties. For example, imagine a group of people at a conference where each person starts by moving around independently, avoiding clustering too early, thus exploring different topics or discussions without being influenced by others.

During the Alignment phase, depicted in Figure 4.1(b), the entities begin to align themselves with others based on commonalities or shared behaviors. This phase is illustrated by the birds adjusting their positions to align with their neighbors, which is quantified by the numbers on the lines (e.g., 0.2, 0.4, 0.8, 0.9, etc.). These numbers represent the degree of alignment or similarity between entities. For instance, in a workplace setting, colleagues may

begin to work more closely together based on shared projects or goals, aligning their efforts to be more cohesive as a team.

Finally, in the Cohesion phase shown in Figure 4.1(c), the entities, which were once separated, come together to form tightly knit clusters or communities. The birds are now grouped into distinct clusters. The connections within each group are stronger, indicating a high level of interaction and mutual influence. For example, in an online community, this would be the stage where users with similar interests form distinct groups, such as forums or chat groups, where they engage more deeply with one another, sharing ideas and collaborating closely.

4.3 Relevant Definitions

Let a graph $G=(V,E)$ represents the given network, which is an undirected and unweighted graph with a set of vertices, V and the set of edges, E. Several symbols were formulated in the proposed technique based on specific definitions. Figure 4.2 illustrates a toy network to explain each definition in detail. This network consists of 23 vertices, divided into two distinct modules, *A* and *B*. The vertices are organized into three levels based on their connectivity. Within each module, the vertices are represented by different shapes to indicate their varying levels of connectivity and their roles in bridging the two modules. This layered structure of level connection with circles representing core vertices, triangles indicating the highest connectivity and influence, squares signifying significant internal connections, and diamonds denoting peripheral connections provide a visual explanation of how different vertices contribute to the network's overall organization and module attractiveness.

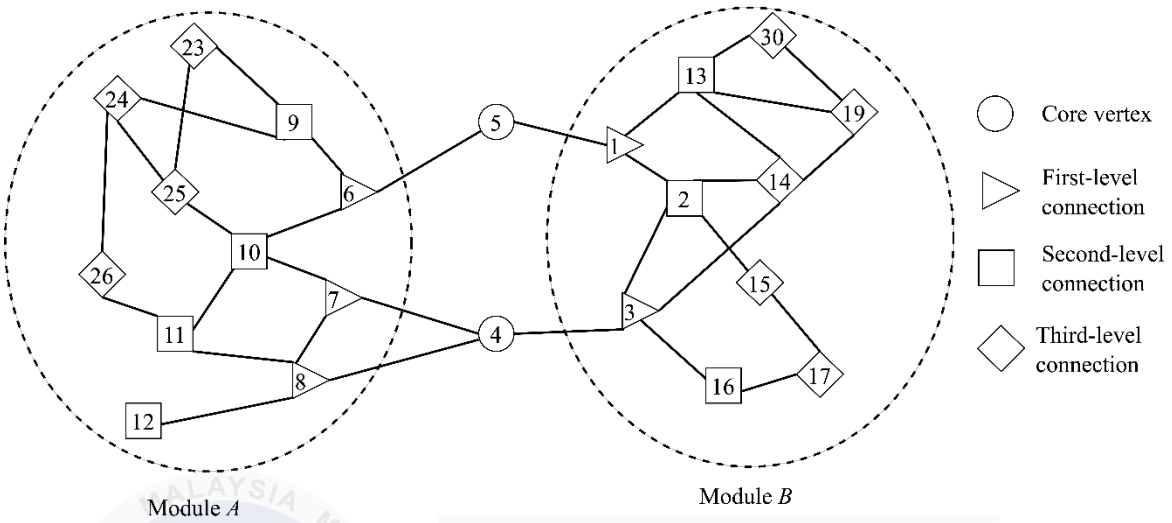


Figure 4.2 A toy network of of two distinct modules, *A* and *B*

4.3.1 Resource Allocation for Similarity Measure

Resource allocation (RA) similarity measures were introduced by Zhou et al. (2009), as represented by the formula in Equation (2.7). Theoretically, the RA index removes the term logarithm used in Adamaic Adar (AA) and instead applies a direct inverse-degree weighting, 1/D(u). This reflects the resource-flow assumption, where a common neighbor with few connections allocates a larger proportion of its limited "resource" to the vertices it connects. As a result, RA imposes a stronger penalty on high-degree vertices and amplifies the contribution of low-degree, exclusive neighbors. This makes RA more sensitive and discriminative in sparse or dynamic networks, where small-degree vertices play a more meaningful role in community cohesion, whereas AA's logarithmic smoothing produces a softer, less aggressive differentiation between hub and non-hub neighbors.

The edge attraction between vertices reflects their similarity, calculated using the *RA* measure, which prioritizes giving resources to high-degree vertices, leading to accurate

but less diverse spreading. Initially, each vertex has its own position based on similarity, and as the bird flock spreads during the alignment phase, modules are naturally revealed by these similarities. This approach is particularly effective in network analysis, as it shows that neighbors with fewer connections significantly impact vertex similarity, making it a valuable method for uncovering structural patterns. Despite being simple, local methods like *RA* are preferred for their efficiency and effectiveness in comparative studies.

From Figure 4.2, the $RA_{4,7}$ similarity between vertices 4 and 7 in the graph can be calculated as shown in Equation (2.7). $RA_{4,7}$ represents the similarity between vertices 4 and 7. The process begins by identifying the neighbors of both vertices. Vertex 4 is connected to vertices 3, 7, and 8, while vertex 7 is connected to vertices 10, 8, and 4. The next step is finding the common neighbors between these two vertices, specifically identifying the vertices they both share. In this case, the common neighbor between vertices 4 and 7 is vertex 8.

Once the common neighbor is identified, the RA similarity is calculated by summing the inverse of the degrees of the common neighbors. The degree of a vertex is defined as the number of edges connected to it. Vertex 8, the common neighbor, has a degree of 4, meaning it is connected to four other vertices in the graph. Using Equation (2.7), the inverse of vertex 8's degree is calculated as $RA_{4,7} = \sum_{D_8 \in CN_{4,7}} \frac{1}{|D(8)|} = \frac{1}{4} = 0.25$. This score reflects the structural similarity between vertices 4 and 7 based on their shared neighbor and the degree of that neighbor.

The details of the calculation for RA similarity between vertex 4 and 7 ($RA_{4,7}$).

Step 1: Identify the *CN*;

- Vertex 4 is connected to the following vertices; 3, 7, and 8.
- Vertex 7 is connected to the following vertices; 10, 8, and 4.

Step 2: Common neighbors between Vertex 4 and Vertex 7: Vertex 8.

Step 3: Calculate RA using Equation (2.7), $RA_{4,7} = \sum_{D_8 \in CN_{4,7}} \frac{1}{|D(8)|} = \frac{1}{4} = 0.25$.

A high similarity score implies that two vertices have strong structural cohesion, which is crucial for forming stable communities. This same similarity calculation was applied uniformly across all vertex pairs in the graph, ensuring a consistent metric for determining attraction between vertices. In dynamic community detection, these scores serve as a key factor in module attraction, continuously influencing how vertices are grouped and how communities evolve over time. The evolving similarity between vertices allows the technique to dynamically adjust community membership, reflecting the current state of the network.

4.3.2 Vertex Attraction

In a graph, $G = (V, E)$, the attractiveness of vertex v to u is represented by the formula in Equation (2.8). Similarly, just as the first bird in a flock can attract others to join and fly together, we introduce a new concept called module attractiveness. This concept explains how a module can attract a vertex based on how well that vertex is connected within the module. The stronger the connections a vertex has within a module, the more attractive

that module becomes to the vertex. Consider an example to calculate the vertex attraction from vertex 4 to vertex 7, $VA_{7\to4}$.

The detailed step of the vertex attraction calculation is as below.

Step 1: Identify the $RA_{4,7}$=0.25 based on the previous calculation.

Step 2: Identify the $D(4) = 3$.

Step 3: Calculate the $VA_{7\to4} = RA_{4,7} * |D(4)| = (0.25) * (3) = 0.75$.

Based on the calculations provided, the vertex attraction between the pairs (4,7) and (8,4) shows differing strengths of connection. For the pair (4,7), the attraction value $VA_{7\to4}$ was 0.75, whereas similar calculation for the pair (8,4), the attraction value $VA_{4\to8}$ was 1.332. This indicates that the connection or attraction from vertex 4 to vertex 8 is stronger than that from vertex 7 to vertex 4. This difference suggests that vertex 4 is more strongly attracted to vertex 8 than to vertex 7, which could imply that in the context of community detection, vertex 4 is more likely to be associated with vertex 8 in the same community, or that the influence of vertex 8 on vertex 4 is stronger than that of vertex 7. The same steps can be applied to compute vertex attraction for different pairs of vertices.

4.3.3 Module Attraction

Theoretical studies of network topology show that structural communities typically form within a radius of approximately 2 to 3 level connection or hops, where vertices share common neighbourhoods, interaction histories, and cluster boundaries. Incorporating the third-level neighbourhood therefore aligns with the natural propagation distance at which community identity becomes most distinguishable. Extending beyond this radius begins to

blend influence from multiple communities, whereas restricting analysis to one- or two-level connections limits the ability to resolve ambiguous or unstable boundary vertices.

In a graph, $G=(V,E)$, the concept of module attraction (MA), is defined as in Equation (4.1).

$$MA_{v\to A} = D_A(u)^2 + \sum_{v\in N(u),v\in A} D_A(v) + \sum_{w\in N(v),w\in A} D_A(w) \tag{4.1}$$

First-level Connection | Second-level Connection | Third-level Connection

where $MA_{v\to A}$ represents the module attraction of a vertex v to a module A by considering the three components: (i) $D_A(u)$ represents the degree of vertex u in the first-level connection, indicating its direct connections of core vertex within the module. (ii) $D_A(v)$ represents the internal degree of vertex v in module A at the second-level connection. Based on the connection between vertices and distinct modules, the $MA_{v\to A}$ can be divided into two conditions: (i) $D_A(v)$ equality, and $D_A(v)$ inequality. (ii) $D_A(w)$ represents the internal degree of vertex w that is a neighbor of the neighbors of u in module A at the third-level connection.

Equation (4.1) can be clarified by calculating the module attraction from first-, second- and third-level of connectivity in the modules. There are two conditions**. (i) First condition, when a vertex is equal within two modules at the first-level connection.** For example, for vertex 5 in Figure 4.2, where a vertex is equal within two modules at the first-level connection for vertex 5 towards Module A and Module B. $D_A(5)=D_B(5)=1$, it is not

easy to move with module, *A* or *B*, is more attracted to vertex 5. Consequently, the impact of indirect neighbors that appears in vertex 5 must be attentively considered.

The calculation for module attraction of vertex 5 to Module *A* ($MA_{5\to A}$).

Step 1: Identify the level connection and degrees.

- First-level connection:
  - Vertex 5 is directly connected to the following vertex; 6.
  - The degree of vertex 5, $D_A(5)=1$.
- Second-level connection:
  - The vertices directly connected to Vertex 6 are; 9, and 10.
  - The degree of vertex 6, $D_A(6)=2$.
- Third-level connection:
  - The vertices connected to the second-level vertices are; 23, 24, 25, 11, and 7.
  - For Vertex 9, with a degree $D_A(9)=2$
  - For Vertex 10, with a degree $D_A(10)=3$.

Step 2: Substitute the values in Step 1 in Equation (4.1), the module attractiveness formula for $MA_{5\to A}$:

$$\begin{aligned} MA_{5\to A} &= D_A(5)^2 + \sum_{v\in N(5)\cap A} D_A(v) + \sum_{w\in N(v)\cap A} D_A(w) \\ &= \left[D_A(5)\right]^2 + \left[D_A(6)\right] + \left[D_A(9)+D_A(10)\right] \\ &= \left[1\right]^2 + \left[2\right] + \left[2+3\right] \\ &= 1^2+2+5=8 \end{aligned}$$

The module attractiveness for Vertex 5 in Module *A* is 8. This value reflects how strongly Vertex 5 is connected within Module *A*.

Meanwhile, the calculation for module attractiveness of vertex 5 to Module *B* ($MA_{5\to B}$).

Step 1: Identify the level connection and degrees.

- First-level connection:
  - Vertex 5 is directly connected to the following vertex; 1.
  - The degree of vertex 5, $D_B(5)=1$.
- Second-level connection:
  - The vertices directly connected to Vertex 1 are; 13, and 2.
  - The degree of vertex 1, $D_B(1)=2$.
- Third-level connection:
  - The vertices connected to the second-level vertices are; 30, 19, 14, 15, and 3.
  - For Vertex 13, with a degree $D_B(13)=3$
  - For Vertex 2, with a degree $D_B(2)=3$.

Step 2: Substitute the values in Step 1 in Equation (4.1), the module attraction formula for $MA_{5\to B}$:

$$\begin{aligned} MA_{5\to B} &= D_B(5)^2+\sum_{v\in N(5)\cap B} D_B(v)+\sum_{w\in N(v)\cap B} D_B(w) \\ &=\left[D_B(5)\right]^2+\left[D_B(1)\right]+\left[D_B(13)+D_B(2)\right] \\ &=\left[1\right]^2+\left[2\right]+\left[3+3\right] \\ &=1^2+2+6=9 \end{aligned}$$

The module attraction for Vertex 5 in Module *B* is 9. This value reflects how strongly Vertex 5 is connected within Module *B*. This is because module *B* is more attractive to Vertex 5, Vertex 5 is more likely to join module *B*.

(ii) Second condition, when a vertex is inequal within two modules at the first-level connection. For example, for vertex 4 in Figure 4.2, where a vertex is not equal within two modules at the first-level connection for vertex 4 towards Module *A* and Module *B*. $D_A(4) \neq D_B(4)$, of which the degree of vertex 4 in module *A* is $D_A(4) = 2$, while the degree of vertex 4 in module *B* is $D_B(4) = 1$. From this value, module *A* is more attractive to vertex 4, then vertex 4 is more likely to join module *A*. It is easy to move to module *A* at first place. However, to clarify using Equation (4.1), the module attraction of vertex 4 to module *A* is $MA_{4\to A} = D_A(4)^2 + \left[D_A(7) + D_A(8)\right] + \left[D_A(10) + D_A(11) + D_A(12)\right] = 2^2 + 3 + 5 = 12$, while the module attraction of vertex 4 to module *B* is $MA_{4\to B} = D_B(4)^2 + \left[D_B(3)\right] + \left[D_B(2) + D_B(14) + D_B(16)\right] = 1^2 + 3 + 5 = 9$. It was proven that module *A* was more attractive to vertex 4, vertex 4 was more likely to join module *A*.

In conclusion, the module attraction analysis for vertex *i* reveals that it has a stronger connection to a module with the highest total attractiveness score. This indicates that vertex *i* is more closely integrated within the module, exhibiting higher connectivity across all three layers, suggesting that it plays a central role in the module's structure.

4.4 Inspired Bird Flock Effect Model for Community Detection

Based on the relevant definitions in Section 4.2, the bird flock effect model is constructed as mentioned in Section 3.2.2.3, Phase 3: Proposed Dynamic Community Detection Inspired by the Bird Flock Effect. The model consists of three rules embedded into the process of community detection in dynamic networks: (1) Rule 1, Separation, is associated with similarity analysis (Part 1: Network Initialization), (2) Rule 2, Alignment, is associated with vertex attraction (Part 2: Submodule Formation), and (3) Rule 3, Cohesion,

is associated with module attraction (Part 3: Inspired by the Bird Flock Effect). The flowchart of community detection inspired by the bird flock effect model is shown in Figure 4.3. The explanation of flowchart of community detection inspired by bird flock effect is provided below:

Part 1: **Network Initialization**. The process begins with the network initialization, where the similarity measure (RA) between vertices is calculated, providing a foundation for understanding the relationships within the network. It is followed by calculating the vertex attraction, which determines the potential of a vertex to either remain within its current submodule or join a new one. This part corresponds to the separation phase in the bird flock effect, where individual vertices (analogous to birds) begin to separate from each other based on their unique characteristics using degree of vertices.

Part 2: **Submodules Formation**. The flow then transitions into submodule formulation, which is associated with the alignment phase. Bird flocking in the sky is not dependent and each bird has their own characteristics and types. Therefore, the differences among the birds show the attractiveness of themselves to others and we call it as vertex attractiveness (cf. Equation (2.8)). The more attraction the birds have, the other birds will join the module or group automatically. In simple terms, every vertex selects its neighbours with the strongest attraction according to two conditions as in Equation (4.2):

$$m_{sub\_u} = \begin{cases} g(u) & , D(u) > D_{\max}(v),\ v \in N(u) \\ \underset{\max(MA_{(u \to v)})}{g(v)} & , D(u) \le D_{\max}(v), v \in N(u) \end{cases} \tag{4.2}$$

where $g(u)$ represents the original module to which vertex u belongs, $D(u)$ is the degree of vertex u, and m_{sub_u} denotes as the submodule. If vertex u has more attraction than its neighbors, then it has more degree and vertex u will remain in the original module, $m(u)$.

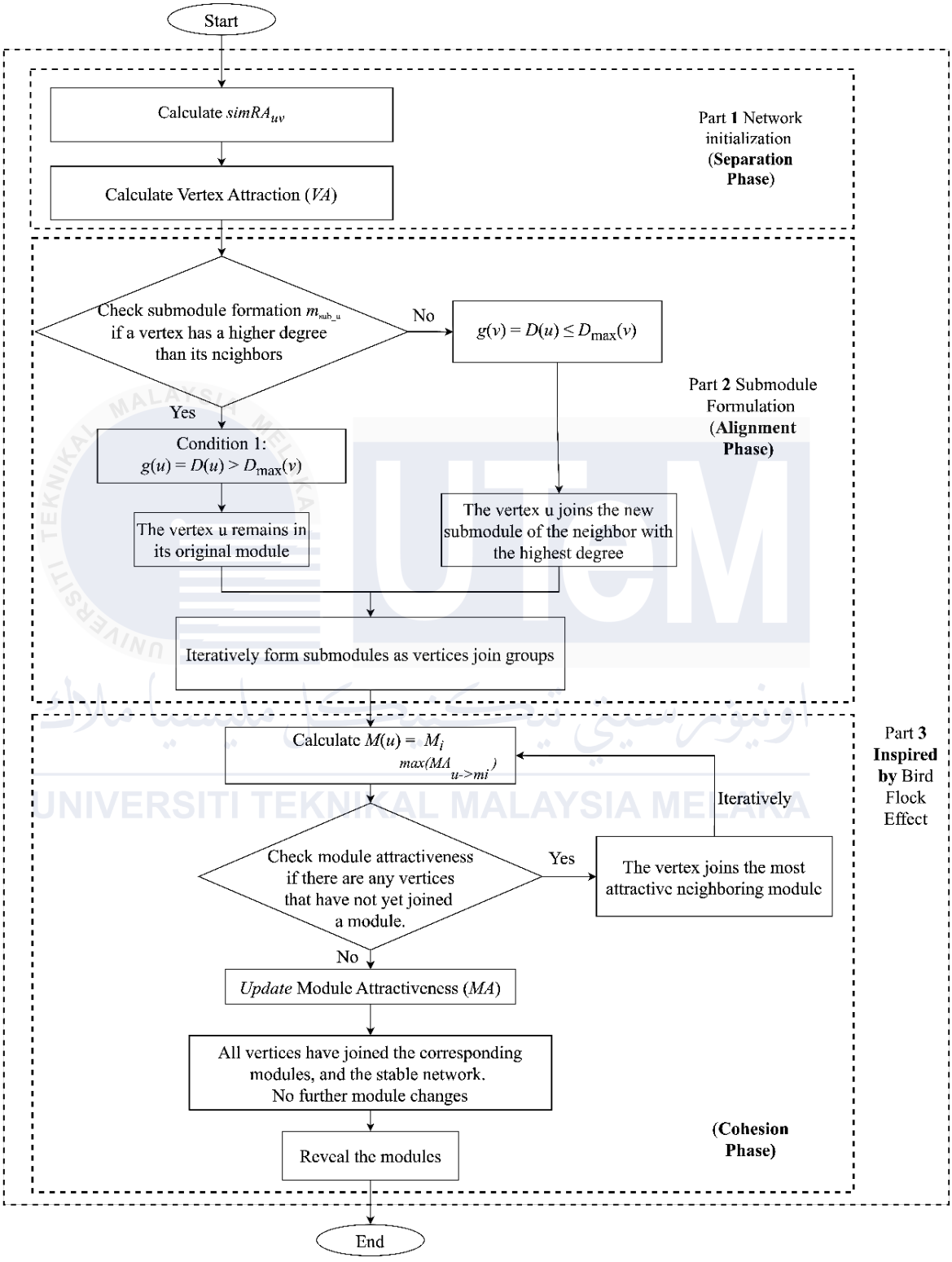


Figure 4.3 Flowchart of community detection inspired by bird flock effect model

Similarly, if neighbor vertex v has more degree, then vertex u will join the module in which vertex v belongs $g(v)$. During this phase, the technique checks whether each vertex has a higher degree of connectivity than its neighbors iteratively. If a vertex's degree is higher, it remains in its original module, but if lower, the vertex joins the submodule of the neighboring vertex with the highest degree. This decision-making process mirrors the behavior of birds aligning themselves within a flock based on the direction and movement of those closest to them.

Part 3: **Inspired by Bird flock effect**. After obtaining the submodules, an increasing number of vertices is attracted to different submodules according to Equation (4.3).

$$M(u) = \underset{\max(MA_{u \to m_i})}{M_i} \tag{4.3}$$

where $M(u)$ is the module that vertex u will join, M_i is the neighborhood module of vertex u is evaluating, and $MA_{u \to m_i}$ denotes the maximum module attractiveness that vertex u experiences towards any of the available submodules m_i. The structure and attractiveness of submodules may change when a vertex joins it by variations of module formation. The update process culminates by iteratively refining and stabilizing the network, ensuring all vertices are grouped into the most attractive neighboring modules, resulting in a cohesive and stable network structure. The bird flock effect metaphorically unites these phases, illustrating the dynamic adjustment and final alignment of all vertices into revealing the modules.

This process continues if there are vertices in the network that have yet to join a module. Over multiple cycles, each vertex adjusts its attractiveness based on the evolving network topology. As vertices join different modules, the network structure gradually

stabilizes, reaching a point where vertices no longer change their module assignments. This stabilization ensures that the community structure is cohesive and well-defined, akin to a flock of birds moving together in harmony once their positions within the group are settled.

4.5 Dynamic Community Detection Inspired by Bird Flock Effect

The bird flock effect is embedded within the dynamic community detection technique. A step-by-step overview of the iterative approach employed by DCDBFE for detecting communities in each network time step is provided. Figure 4.4 illustrates the complete framework of the proposed technique. The process begins with constructing the network using either an extended LFR dataset or real-world networks. The proposed dynamic community detection approach consists of two main components: initial community detection and incremental community detection, each designed to identify and adapt the community structure within evolving networks.

The detailed framework of DCDBFE: Start with **initial community detection.** This initial community detection step is labelled as CDBFE. This component is guided by a model inspired by the bird flock effect, which involves three phases: Separation, Alignment, and Cohesion. This approach obtains initial modules M_0 at time step T_0 from the whole network G_0.

By studying the model, researchers can now guess how these modules will react to changes in their surroundings. According to the three phases of bird flock effect model mentioned earlier, Figure 4.5 demonstrates the process of incremental community detection based on the bird flock effect from time steps T_1 until T_8. This figure illustrates how

communities evolve over time within a network as vertices, V and edges, E are added or removed, thereby affecting the overall community structure.

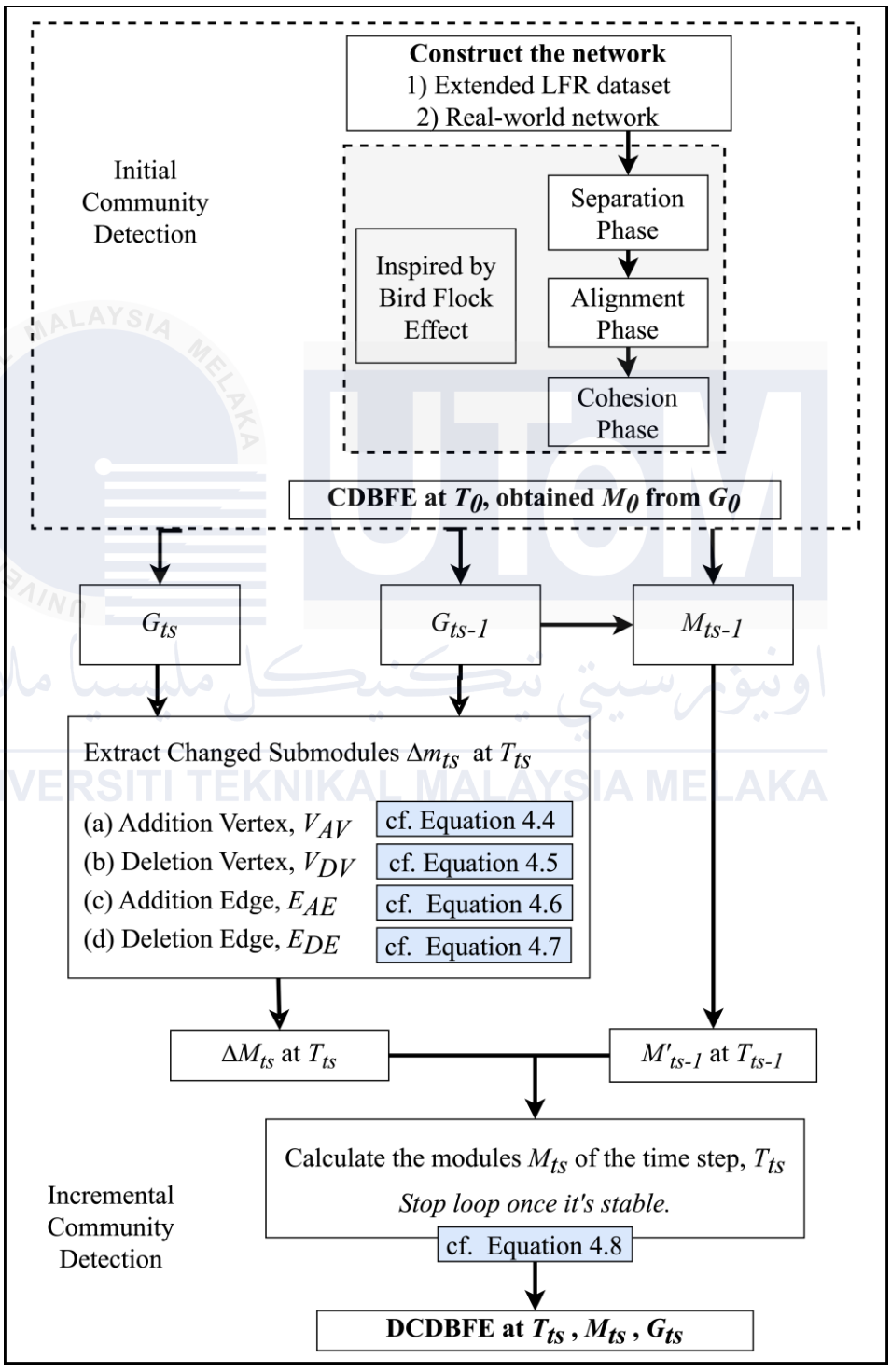
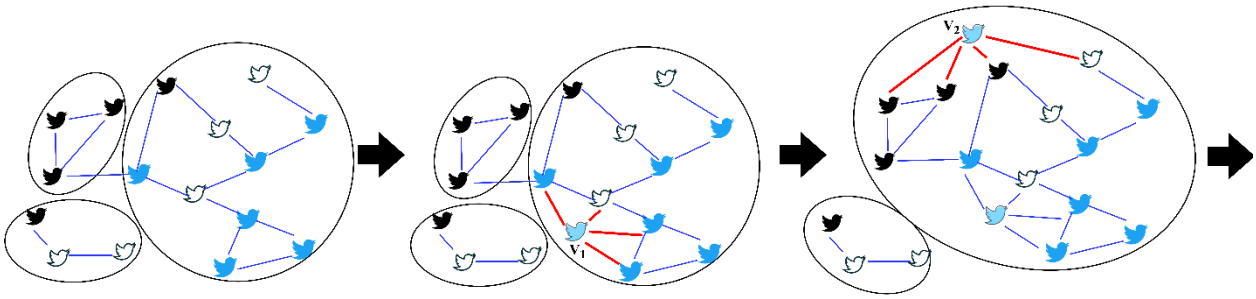
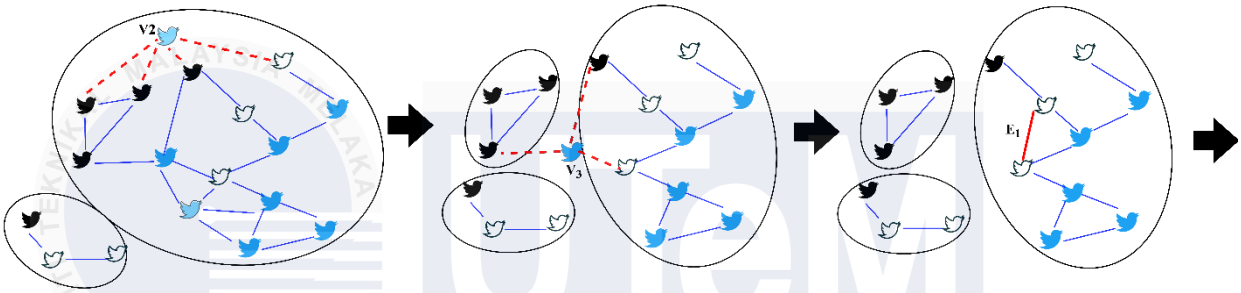


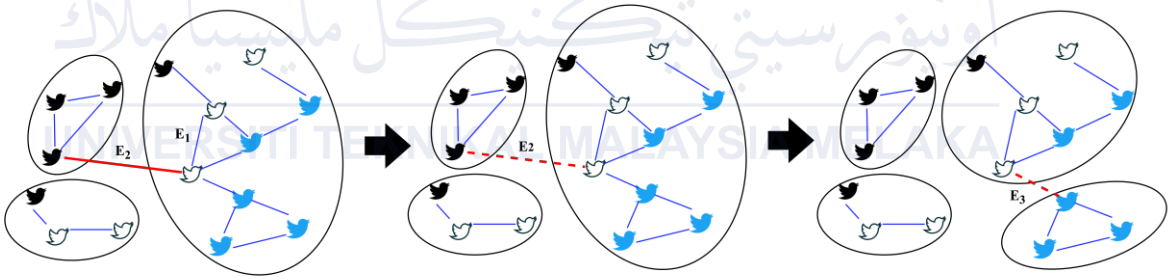
Figure 4.4 Framework of DCDBFE



(a) Initial community detected at T_0 (b) Addition vertex V_1 in a module at T_1 (c) Addition vertex V_2 between modules at T_2



(d) Deletion vertex V_2 in a module at T_3 (e) Addition vertex V_3 between modules at T_4 (f) Addition edge E_1 in a module at T_5



(g) Addition edge E_2 between modules at T_6 (h) Deletion edge E_2 between modules at T_7 (i) Deletion edge E_3 in a module at T_8

Figure 4.5 Demonstration of the process of incremental community detection based on bird flock effect from time steps T_1 until T_8

Incremental community detection. This incremental community detection step, labeled as DCDBFE, will repeat the process of community detection at each time step after the initial community detection. The detailed process of incremental community detection is provided below:

(1) **Extract Changed Submodules,** Δm_{ts}. In contrast to a static network, the vertices and edges in a dynamic network can vary over time, leading to ongoing changes in the community structure across different time steps. These changes include adding or deleting vertices and edges, which are handled by specific techniques.

- **Addition Vertex,** V_{AV} . When a new vertex and its associated edges are added to the current time step, G_{ts} , rather than the previous time step, G_{ts-1} it is referred to as a vertex addition event. Let's denote the addition vertex as V_{AV} . We define this formally in Equation (4.4):

$$V_{AV} = \{u \mid u \in V_{ts},\ u \notin V_{ts-1}\} \tag{4.4}$$

where V_{ts} and V_{ts-1} denote the vertices in networks G_{ts} and G_{ts-1}, respectively. The addition of new vertices can lead to changes in the community structure within the network, and two possible scenarios may arise: (i) the addition of a vertex within a single module, and (ii) the addition of a vertex between two modules. Figure 4.5(b) demonstrates when a vertex V_1 is introduced within a module at T_1, the community structure remains unchanged. The inclusion of additional vertices within a module enhances interconnectivity, consequently increasing the connection density. However, the overall number of modules remains unchanged, and the newly inserted vertex must be partitioned within the community. Quick Community Adaptation (QCA) provides theoretical proof for this (Nguyen et al., 2011). On the other hand, Figure 4.5(c) shows the addition of a vertex V_2 between two modules at T_2 has the potential to induce modifications in the community structure. In this scenario, it is necessary to document the newly added vertex along with its corresponding edges

and the modules it connects to. These elements are then incorporated into the submodules, denoted as Δm_{ts}.

- **Deletion Vertex, V_{DV}.** When a vertex and its associated edges are eliminated from the current time step G_{ts}, rather than the previous time step G_{ts-1}, it is referred to as a deletion vertex event. Deletion vertex is represented by V_{DV} as shown in Equation (4.5):

$$V_{DV} = \{u \mid u \notin V_{ts},\ u \in V_{ts-1}\} \tag{4.5}$$

where V_{ts} and V_{ts-1} denote the vertices in networks G_{ts} and G_{ts-1}, respectively. Figure 4.5(d) and Figure 4.5(e) show a deletion edge within single module and between modules, respectively. We observe that deletion vertex of vertex V_2 within a module at T_3 and vertex V_3 between modules at T_4 led to changes in the community structure. When deleting a vertex, its associated edges must also be removed. The connected modules are then added to the submodules Δm_{ts}.

- **Addition Edge, E_{AE}.** Similarly, adding new edges to the current time step G_{ts}, as opposed to the previous time step G_{ts-1} is referred to as an edge addition event. This is defined formally in Equation (4.6):

$$E_{AE} = \{e \mid e \in E_{ts},\ e \notin E_{ts-1}\} \tag{4.6}$$

where E_{ts} and E_{ts-1} denote the edges of networks G_{ts} and G_{ts-1}, respectively. Adding edge E_1 inside a module at T_5 will not change the structure of the community; instead, it will increase the density of edges within the module as shown in Figure 4.5(f). However, Figure 4.5(g) shows changes in the community structure

may result from the addition of edges E_1 between modules at T_6. Therefore, there is no need to re-detect the module for newly added edges.

- **Deletion Edge, E_{DE}.** An edge deletion event occurs when an edge is eliminated from the previous time step G_{ts-1}, and instead appears in the current time step G_{ts}. The deleted edges are represented by E_{DE} as shown in Equation (4.7):

$$E_{DE} = \{e \mid e \notin E_{ts},\ e \in E_{ts-1}\} \tag{4.7}$$

where E_{ts} and E_{ts-1} denote the edges of networks G_{ts} and G_{ts-1}, respectively. A community's structure may change as a result of edge deletion. However, because there are typically few connections between modules, eliminating the edges E_1 between them at T_7 often has little effect on their structure. In contrast, Figure 4.5(i) shows that removing edge E_3 within a module at T_8 might cause a split or division. Consequently, it is necessary to compute the deleted edges and the related modules in the submodules Δm_{ts}.

(2) **Compute Changed Modules ΔM_{ts}.** By adding and deleting vertices and edges from the network, we may compute the submodules Δm_{ts}, which may change the community structure. Next, we used the bird flock effect model to find modules in the submodules to complete Algorithm 4.1 (CDBFE).

(3) **Compute Unchanged Modules, M'_{ts-1}.** The modules M_{ts-1} can be obtained at time steps G_{ts-1}. Additionally, the modules whose structures are most likely to change, as indicated by the changed submodules Δm_{ts}, can be computed. By

computing the difference between the two sets, the unchanged modules M'_{ts-1} at current time step G_{ts} can be obtained.

(4) **Compute all Modules, M_{ts} of network G_{ts} .** Modules in time step, M_{ts} consist of unchanged modules M'_{ts-1} and changed modules ΔM_{ts} . Let M_{ts} represents the modules of network G_{ts} at time step, ts , which is defined in Equation (4.8):

$$M_{ts} = M'_{ts-1} + \Delta M_{ts} \tag{4.8}$$

In this study, dynamic community detection based on bird flock effect model used repeated event processing such as adding and removing vertices and edges, to produce an accurate and efficient community. In DCDBFE, the updating process at time step, ts , stopped when the changed region of the network reached stability. After identifying the vertices and modules affected by structural events, ΔM_{ts}, the algorithm iteratively updated only these affected vertices using the module-attraction rule until no vertex changed its module assignment between two consecutive iterations. If $\Delta M_{ts} = \varnothing$, the algorithm immediately terminates and sets, $M_{ts} = M'_{ts-1}$, since no structural changes occurred. To ensure convergence, two safeguards were applied: an oscillation check, where a vertex cycling between modules is resolved by selecting the assignment with higher total attraction strength, and a maximum-iteration limit to prevent excessive looping. Once stability is reached meaning no further changes are observed, the final module set for the current time step is produced as $M_{t_s} = M'_{t_{s-1}} + \Delta M_{t_s}$.

4.6 DCDBFE Pseudo Code

In this section, the DCDBFE pseudocode is introduced as an extension of CDBFE, specifically proposed for identifying communities in dynamic networks. While CDBFE is designed for static networks, it has been improved and modified to operate effectively in a dynamic setting. The DCDBFE consists of three main steps: (1) Initial community detection, (2) Submodule adjustment, and (3) Dynamic community detection. The pseudocode is presented in Algorithms 4.1–4.6, with a detailed explanation of the steps involved in each algorithm.

Initialization Community Detection. The bird flock effect model was applied to simulate the network's behaviour in detecting communities, drawing inspiration from how birds flock together based on their positions and attractions to one another. Initially, each vertex was treated as a separate module, with its degree representing its resources, following Equation (2.7). Submodules were then formed by drawing vertices towards their neighbouring vertices, as described in Equation (2.8). The bird flock effect was further simulated through an iterative process, following the procedures outlined in Equations (4.1) and (4.3). This iterative approach is crucial in Phase 3, where module attraction was calculated using Equation (4.1). The process involved evaluating each vertex's attraction to its neighbours, extending up to the third level of connection. Equation (4.2) models the structure and attractiveness of submodules for each vertex. The network's module structure reached stability when all vertices joined different modules, and no further changes occurred.

If the community structure of the network is known at advance, ground truth measure is used (Bouhatem et al., 2021). The most popular measurement for evaluating the quality of community detected uses the normal mutual index (*NMI*) (Danon et al., 2005) and

adjusted random index (*ARI*) (Rand, 1971) as metric. Due to topologically driven influences, the network eventually converges as it changes over time, allowing us to obtain the optimal module partition.

For example, flock control is distributed to each bird by enabling them to follow their neighbors' actions. The behavior of separation requires birds to avoid their neighbors, with the force of separation increasing as they come closer. Due to this separation rule, each bird in the flock creates space around itself, allowing for greater maneuverability, reducing the risk of collisions. A balanced flock is achieved when all separation forces equal to zero, indicating that each bird is sufficiently distanced from its neighbors. Birds coordinate their direction based on similarity values, with greater attraction helping to balance the flock as birds maintain the same direction, preserving both separation and cohesion forces. Consequently, the CDBFE was employed to determine the community structure of the original network at a given time step, T_0. Algorithm 4.1 provides the pseudocode for CDBFE.

Algorithm 4.1 CDBFE

Input: $G(t) = (V, E)$
Output: Final set of modules, $M(v)$

1://**Initialize communities of each vertex in changed submodule**
2: **for** each vertex v in V **do**
3: $M(v) = N(v)$ the number vertex of v
4: **end for**
5://**Compute the submodules**
6: **for** each vertex v in V do
7: **for** each vertex u in $N(v)$ **do**
8: compute the **RA similarity measure** using Equation (2.7)
9: compute the **vertex attraction** using Equation (2.8)
10: Imax = 0
11: **while** Flag do

12: compute submodules using Equation (4.2)
13: **end for**
14://**Simulate the bird flock effect**
15: Flag = TRUE
16: NMImax = 0
17: **while** Flag do
18: **for** each vertex v in V **do**
19: **for** each vertex u in $N(v)$ **do**
20: compute the **module attraction** using Equation (4.1)
21: **end for**
22: simulate the bird flock effect using Equation (4.3)
23: **end for**
24: S = NMI (C; true clusters)
25: if S > NMI max then
26: NMI max = S
27: **else**
28: Flag = FALSE
29: **end if**
30: **end while**
31: //**return modules** $M(v)$ **in changed submodule**

• **Changed Submodules.** In considering actions that may lead to changes in community structure, network events were categorized into four types: vertex addition, vertex deletion, edge addition, and edge deletion. Algorithms 4.2–4.5 illustrate the detailed process by which each event returns a changeable submodule, Δm_{ts}.

Algorithm 4.2 Addition Vertex Event (cf. Equation (4.4))

Input: V_{AV}, G_{ts}, M_{ts-1}

Output: Changed submodules, Δm_{ts}

1: **for** each vertex $v \in V_{AV}$ **do**
2: **if** $N(v)$ in the same module $M_{N(v)}$ **then**
3: $M_{N(v)} \leftarrow v$
4: **else**
5: $\Delta m \leftarrow v$
6: **for** each vertex $u \in N(v)$ **do**
7: $\Delta m_{ts} \leftarrow M(u)$
8: **end for**
9: **end if**
10: **end for**

Algorithm 4.3 Deletion Vertex Event (cf. Equation (4.5))

Input: V_{DV}, G_{ts}, M_{ts-1}

Output: Changed submodules, Δm_{ts}

1: **for** each vertex $v \in V_{DV}$ **do**
2: **for** each vertex $u \in N(v)$ **do**
3: $\Delta m_{ts} \leftarrow M(u)$
4: **end if**
5: **end for**

Algorithm 4.4 Addition Edge Event (cf. Equation (4.6))

Input: E_{AE}, M_{ts-1}

Output: Changed submodules, Δm_{ts}

1: **for** each edge $e \in E_{AE}$ $e \in E_{AE}$ **do**
2: **if** $M(u) \neq M(v)$ **then**
3: $\Delta m_{ts} \leftarrow M(u)$
4: $\Delta m_{ts} \leftarrow M(v)$
5: **end if**
6: **end for**

Algorithm 4.5 Deletion Edge Event (cf. Equation (4.7))

Input: E_{DE}, M_{ts-1}

Output: Changed submodules, Δm_{ts}

1: **for** each edge $e \in E_{DE}$ **do**
2: **if** $M(u) = M(v)$ **then**
3: $\Delta m_{ts} \leftarrow M(u)$
4: **end if**
5: **end for**

• **Dynamic Community Detection.** Over time, communities were identified by building upon the obtained changed submodules using the bird flock effect model. The DCDBFE, as outlined in Algorithm 4.6, began with an initial community detection at time step, T_0. In Step 1, the technique received the dynamic network, represented as a series of time steps d $DynG_t = \{G_0, G_1, ..., G_{ts}\}$, as input. Step 2 involves applying the (CDBFE) function (Algorithm 4.1) to the initial network time step, G_0, which resulted in the first set of modules

M_0. This set M_0 served as the baseline for further dynamic community detection in subsequent time steps. The dynamic phase of the dynamic community detection began at Step 3, where a loop iterated through each time step starting from $ts = 1$.

Within each iteration, Step 5 computed the sets of vertices and edges that had been added or deleted at the current time step, denoted as $V_{AV}, V_{DV}, E_{AE}, E_{DE}$. In Steps 6 and 7, the algorithm updated the community structure by adding and deleting vertices using the respective equations (cf. Equations 4.4 and 4.5). Steps 8 and 9 involved updating the edges in the community structure based on the added and deleted edges using equations (cf. Equations 4.6 and 4.7). In Step 10, the algorithm calculated the modules that remained unchanged from the previous time step. Step 11 then applying the CDBFE function to the modified network time step ΔG_{ts}, producing an updated module structure M_{ts}. Finally, Step 12, computing the complete module or community structure for the current time step using Equation 4.8, and the loop continued until all time steps were processed, concluding in Step 13.

Algorithm 4.6 DCDBFE

Input: $DynG_t = \{G_0, G_1, ..., G_{ts}\}$

Output: $DynM_t = \{M_0, M_1, ..., M_{ts}\}$

1: //Initial community detection at time step, T_0
2: $M_0 = CDBFE(G_0)$ # using Algorithm 4.1 (CDBFE)
3: //Dynamic Dynamic community detection at time step, ts
4: **for** $ts = 1$ to **do**
5: compute $V_{AV}, V_{DV}, E_{AE}, E_{DE}$ using Equation (4.4) – (4.7)
6: $\Delta m_{ts} \leftarrow$ Addition Vertex $(V_{AV}, G_{ts}, M_{ts-1})$ using Equation (4.4)
7: $\Delta m_{ts} \leftarrow$ Deletion Vertex $(V_{DV}, G_{ts-1}, M_{ts-1})$ using Equation (4.5)
8: $\Delta m_{ts} \leftarrow$ Addition Edge (E_{AE}, M_{ts-1}) using Equation (4.6)
9: $\Delta m_{ts} \leftarrow$ Deletion Edge (E_{DE}, M_{ts-1}) using Equation (4.7)

10: compute the unchanged modules, M'_{ts-1}
11: $M_{ts} \leftarrow CDBFE(\Delta G_{ts})$
12: Compute M_{ts} using Equation (4.8)
13: **end for**

4.7 Summary

In conclusion, Chapter 4 introduces the preliminary study and details of the proposed technique based on an incremental approach inspired by the bird flock effect to improve the stability of community structures in network analysis. Combining the RA similarity measure and module attraction up to the third level of connections offers a more comprehensive understanding of communities detected over time using the incremental approach.

Further analysis of the proposed technique will be explained in Chapter 5, the experimental evaluation of the proposed technique, analysing its performance across both synthetic and real-world datasets. Lastly, it provides comparative results and statistical validation.

CHAPTER 5

RESULT AND DISCUSSION

5.1 Introduction

This chapter presents a thorough analysis of the conducted research, focusing on three main results of three phases: Phase 1, Phase 2, and Phase 4. The results of Phase 1 introduce the preliminary findings of the similarity analysis, which aim to develop a basic understanding of network topologies and assess the initial performance of the proposed similarity measures. Next, in Phase 2, any necessary methodological changes based on the initial findings are implemented up to the fourth level connection for module attraction, with the goal of achieving the best performance. Finally, the results of Phase 4 focus on the performance evaluation of the proposed technique (DCDBFE) and the validation of these results. The evaluation was conducted by comparing the proposed technique with existing dynamic community detection techniques using benchmark synthetic and real-world network datasets. The results from each phase highlight the stability of the proposed dynamic community detection by validating its performance using statistical approach.

5.2 Result Phase 1: Preliminary Similarity Analysis

In the preliminary analysis, this thesis selectively examined three computationally efficient datasets rather than exhaustively testing all datasets proposed in the literature. These datasets were chosen for their effectiveness in revealing significant insights relevant to community detection. Three distinct datasets were selected to investigate community merging and splitting events. Merger and split events were chosen because they represent

the most challenging and informative changes in dynamic networks. These events test whether the technique can preserve temporal stability when communities combine or divide. These selected datasets are a part of extended LFR network to simulate the variations of community structure to ease the understanding of the concept. In this study, six similarity measures related to specific topological properties, as mentioned in Section 3.2.2.1, were chosen.

The six similarity measures involved were as follows: Jaccard Index, Leicht-Holme-Newman Index (LHN), Salton Index, Sorensen Index, Adamic-Adar Index (AA), and Resource Allocation Index (RA). The impact of each similarity measure is discussed based on the performance evaluation in this section, which employed the Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI) to see the improvement of stability in dynamic community detection.

5.2.1 Performance of Comparison Similarity Measures

This subsection presents a comparative analysis of different similarity measures metrics applied to a network with varying levels of merging and splitting within communities. The merge-split community scenarios were labeled as m5_s5 for 5 merger events and 5 split events, m20_s20 for 20 merger events and 20 split events, and m40_s40 for 40 merger events and 40 split events. The analysis in Phase 1 focused on the first level of connectivity only in module attraction to detect communities as shown in Figures 5.1 to 5.7.

5.2.1.1 Scenario 1: 5 Events of Merger and 5 Events of Split Community

The experiment began with five events involving community merging and splitting. Figure 5.1 presents a comparison of the NMI metric, which evaluated the similarity between the predicted and true community structures. The graph indicated that most similarity measures maintained a high NMI value close to 1 across time steps, proving that the community structures identified by these techniques remain consistently similar to the ground truth. The results showed that the Jaccard, AA, and RA measures consistently achieved higher NMI values, with RA demonstrating the best performance, thereby indicating superior stability in community detection. In contrast, the Salton measure exhibited significant fluctuations, suggesting reduced stability in detecting community structures during merging and splitting events. In time steps 14 and 20, the RA and AA similarity measures exhibited high NMI values, with RA reaching 0.998630 at time step 14 and 0.998629 at time step 20, indicating superior stability in community detection. In contrast, the Salton measure showed significant fluctuations, dropping below 0.9900, which suggests less consistency in detecting communities during these time steps.

Figure 5.2 compares the ARI values across different similarity measures. The ARI metric adjusts for chance and measures the similarity between two data communities, considering all pairs of samples. Similar to the NMI results, Jaccard, AA, and LHN measures maintained relatively high ARI values, demonstrating strong performance in community detection, though slightly less consistent than RA. The Sorensen measure showed moderate stability with some fluctuations, while the Salton measure experienced significant variability, indicating it as the least reliable in preserving community structures during dynamic merging and splitting events.

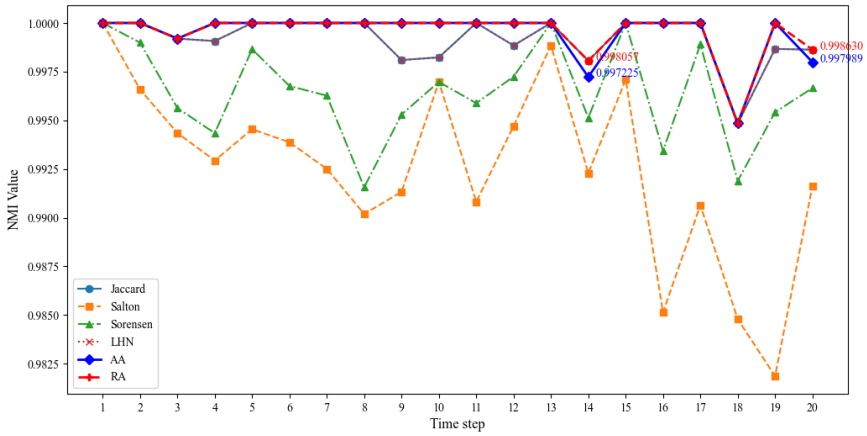


Figure 5.1 Comparison of similarity measures based on NMI metric with 5 events of merger and 5 events of split (m5_s5)

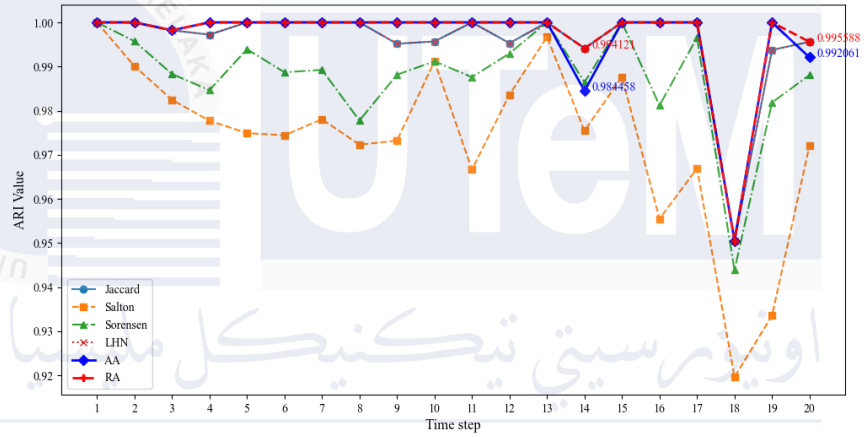


Figure 5.2 Comparison of similarity measures based on ARI metric with 5 events of merger and 5 events of split (m5_s5)

In time steps 14 and 20, the RA and AA similarity measures displayed high ARI values, with RA achieving 0.994121 at time step 14 and 0.995588 at time step 20, indicating stable performance in community detection. In contrast, the AA measure dropped to 0.984458 at time step 14 and 0.992061 at time step 19, showing slight variability but still maintained the overall stability compared to other measures, such as Salton, which exhibited larger fluctuations. The observed numerical difference of 0.000001 is extremely small and falls within the expected floating-point precision tolerance of the computational environment. Such a minimal variation does not represent a practically meaningful improvement; however, it indicates a high level of

stability and reproducibility in the technique's performance, which still contributes to domain knowledge.

5.2.1.2 Scenario 2: 20 Events of Merger and 20 Events of Split Community

Next, the experimental results, illustrated in Figures 5.3 and Figure 5.4, present the performance of various metrics during the Merge-Split process involving 20 merge-split events (m20_s20). In Figure 5.3, the RA similarity measure consistently achieved near-perfect NMI scores close to 1.000, demonstrating exceptional stability in preserving the true community structure. In contrast, metrics like Sorensen and LHN exhibited more variability, with NMI values occasionally dipping below 0.985, indicating a less stable performance.

Figure 5.4 further supports RA's superior performance with the ARI metric, where RA maintained values near 1.000, while other methods, particularly Sorensen and LHN, showed fluctuations and lower ARI values, dropping to around 0.94 at some time steps.

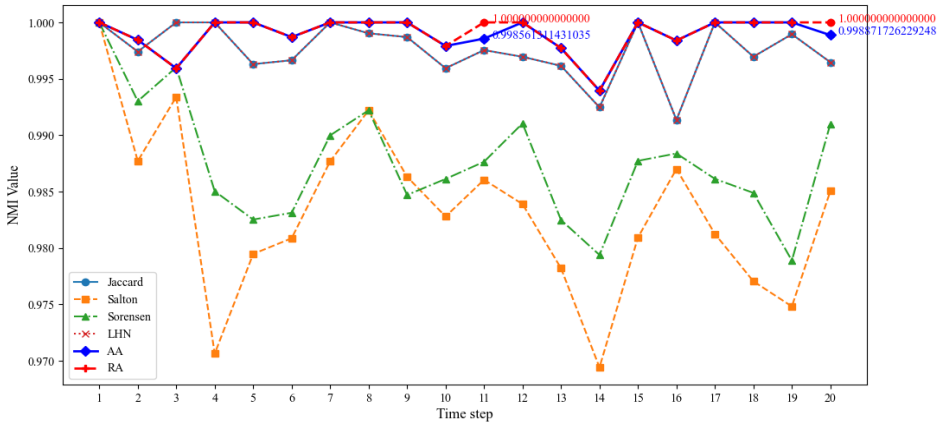


Figure 5.3 Comparison of similarity measures based on NMI metric with 20 events of merger and 20 events of split (m20_s20)

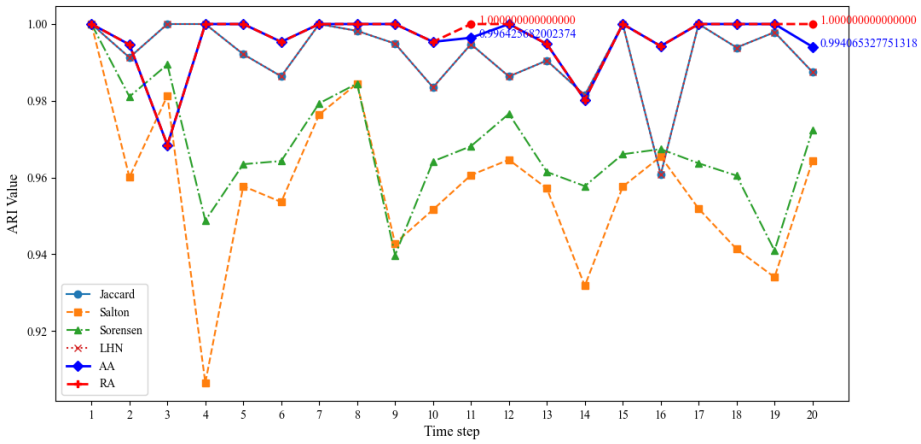


Figure 5.4 Comparison of similarity measures based on ARI metric with 20 events of merger and 20 events of split (m20_s20)

5.2.1.3 Scenario 3: 40 Events of Merger and 40 Events of Split Community

Continuing with the experimental results in Phase 1, Figure 5.5 and Figure 5.6 illustrate the performance of various metrics when the number of merge-split events was increased to 40 (m40_s40). In Figure 5.5, the NMI metric shows that RA still performed well but experienced more noticeable fluctuations compared to previous experiments, particularly around time steps 7.5 and 12.5, where the NMI values dipped significantly for Sorensen and LHN, dropping below 0.975 and even reached around 0.850 at certain points. Despite these dips, RA maintained higher consistency relative to other methods.

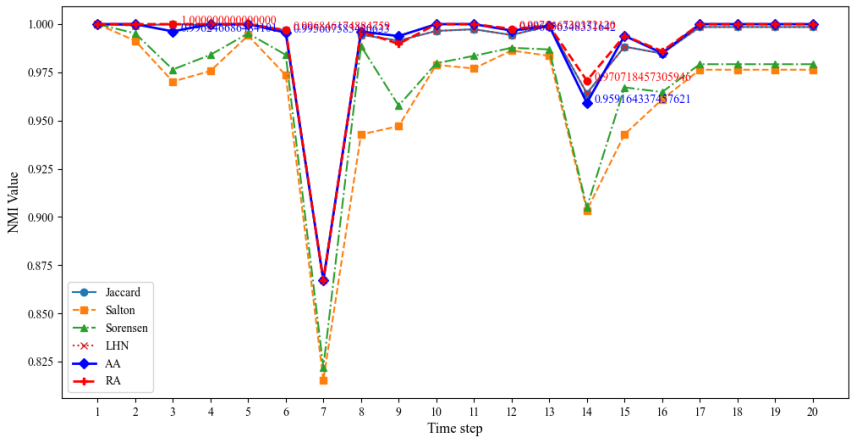


Figure 5.5 Comparison of similarity measures based on NMI metric with 40 events of merger and 40 events of split (m40_s40)

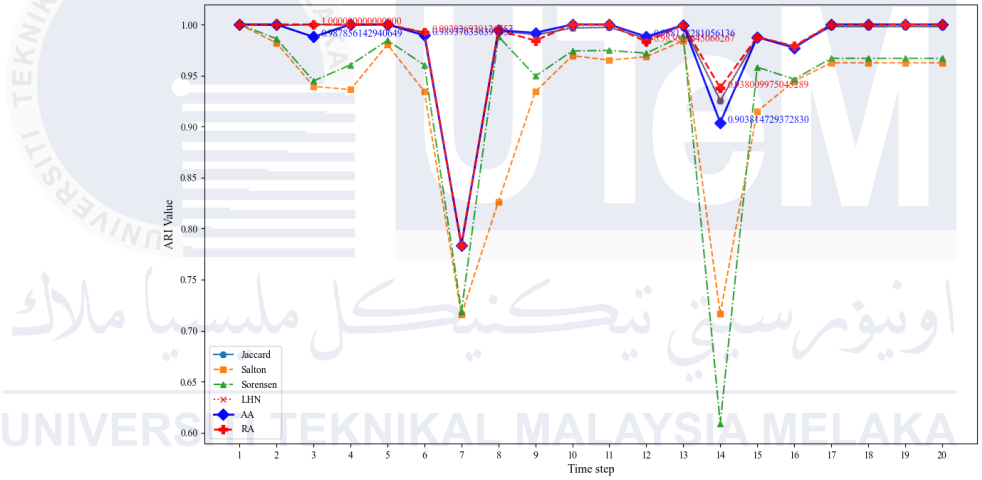


Figure 5.6 Comparison of similarity measures based on ARI metric with 40 events of merger and 40 events of split (m40_s40)

Figure 5.6 further emphasizes these fluctuations with the ARI metric, where RA and other methods like AA experienced a notable dip around the same time steps, reaching as low as 0.600 for LHN and Sorensen. However, RA and AA recovered quickly, maintaining an ARI close to 1.000 for most of the time steps, indicating that they still perform better than the other methods despite the increased complexity.

5.2.1.4 Execution time for the three types of datasets used

Figure 5.7 illustrates the execution time required for preliminary similarity analysis across different similarity measures (AA, RA, Jaccard, LHN, Sørensen, and Salton) under varying levels of network complexity, represented by different numbers of merge-split events (5, 20, and 40 events). The figure clearly shows that as the number of merge-split events increases, the execution time for each similarity measure also increases. The Salton measure exhibited the highest execution time (225 seconds) for the most complex scenario (40 merge-split events), while AA and RA measures were the least computationally expensive, where this study focused on RA. The RA similarity measure demonstrated relatively low computational time compared to other measures, indicating its efficiency in the preliminary similarity analysis. This suggests that the computational burden of similarity analysis scales with both the complexity of the network dynamics and the choice of the similarity measure; with more dynamic networks requiring significantly more processing time, especially for measures like Salton.

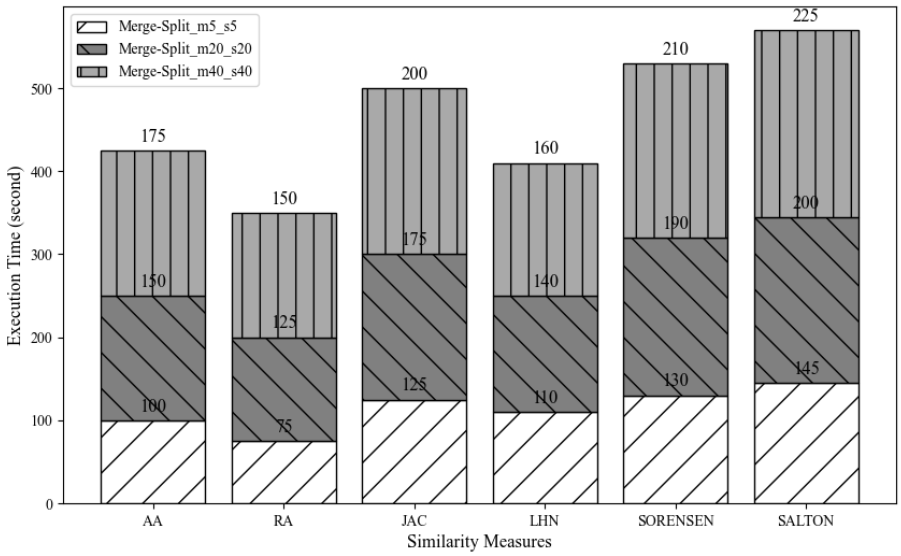


Figure 5.7 Execution time for preliminary similarity analysis

5.2.2 Summary of Results in Phase 1

In conclusion, these preliminary results paint a very promising picture of the community detection's performance. The consistently high values across two different evaluation metrics performances with first levels of network connectivity provide strong evidence for the stability and effectiveness. This multi-faceted evaluation approach strengthens the validity of the findings.

However, it's important to note that these results were based on synthetic networks. While synthetic networks are valuable for controlled testing, the last crucial step would be to validate the performance on real-world networks, which often exhibits more irregular and unpredictable structures.

In the context of dynamic networks, the "ground truth" refers to the known or predefined community structure as explained in Section 2.6.2 that serves as a benchmark for evaluating the performance of community detection. If the ground truth is unstable, it means that the actual community structure itself is subject to frequent changes, even without considering the additional noise or complexity introduced by merge-split events. It inherently becomes more difficult for any community detection to match the detected communities to the true ones, even if the technique is highly accurate. This is because the baseline (ground truth) itself is fluctuating, making the task more challenging.

This instability could be due to inherent properties of the network, such as high turnover rates of vertices within communities, or a rapid and frequent reconfiguration of the network's topology. The challenge can lead to lower and more variable similarity scores, as the techniques struggle to keep up with the fast-paced changes in the community structure.

Across different similarity measures, the RA similarity measures consistently showed the highest levels of similarity, indicating their effectiveness in capturing the network's structure and dynamics. Second highest were AA and LHN similarity measures in which the performances were high and stable. On the other hand, the Salton, Sorensen, and Jaccard methods exhibited lower similarity values, suggesting they might not be as effective in this context. This consistency across only first layer of connectivity suggests that the observed patterns are robust and not just specific to a particular network configuration. To further understand these results, it would be beneficial to explore why certain measures perform better than others and consider additional analysis techniques in the whole community detection.

This could provide deeper insights into the network's properties and helps validate the findings by comparing them with other datasets or established frameworks. Overall, these results provide a solid foundation for further analysis and understanding of the network dynamics. Such comparison highlights the relative stability and effectiveness of these methods in measuring similarity within the network, offering a basis for selecting the most suitable approach for specific applications.

5.3 Result Phase 2: Module Analysis

In this section, the second experiment was conducted in Phase 2 which is to enhance the level of connectivity for module attraction from first to fourth level as explained in Section 3.2.2.2. This was proven by the results in Figures 5.8 and 5.9 that show the results of comparison when fixing the RA similarity measure and changing the level of connectivity layer for the impact module attraction to the community detected. The quality metrics used

were NMI, and ARI. Dataset Merge-Split_m40_s40 was implemented for this module analysis. This is because from a previous research in (Z. Sun et al., 2022), it was stated that almost all the comparison techniques decreased suddenly at the seventh time step. Here, this study intended to solve the unstability when the performance of most techniques fluctuated greatly when the number of merged and split communities reached 40, at which point, almost every community in the dynamic network had changed.

5.3.1 Performance of Comparison Level Connection

Building upon the findings from Phase 1 of this research, where a comprehensive similarity analysis was conducted to identify the most effective method for community detection, the RA similarity measure emerged as the best-performing calculation. This measure demonstrated superior stability in capturing underlying community structures within dynamic networks. Recognizing its potential, this section delves deeper into evaluating the RA similarity measure by experimenting with varying levels of connectivity within synthetic networks. The goal was to assess the performance of the RA measure under different conditions of network complexity, particularly in scenarios where community structures evolve through merging and splitting. By analyzing the RA similarity measure across these different levels of connectivity, this study aimed to further validate its stability and effectiveness in dynamic community detection tasks.

The results depicted in Figures 5.8 and Figure 5.9 represent a comparison of the NMI and ARI metrics, respectively, for different level connections in synthetic networks, specifically within the 40 events of merger and 40 events of split community dataset (m40_s40). Both figures evaluate the performance of the community detection across four

levels of connections (First, Second, Third, and Fourth Level) using a fixed RA similarity measure.

Community mergers happen when increasing inter-community edges or shared vertices blur the boundaries between groups, while splits occur when internal cohesion decreases or subgroups form. These dynamics were simulated through the merger, m and split, s parameters in the synthetic datasets. The selected experiment maintained stability during these events because its similarity components and module-attraction preserved temporal continuity even when the network structure changed.

5.3.1.1 Scenario 4: 40 Events of Merger and 40 Events of Split Community using RA Similarity Measure

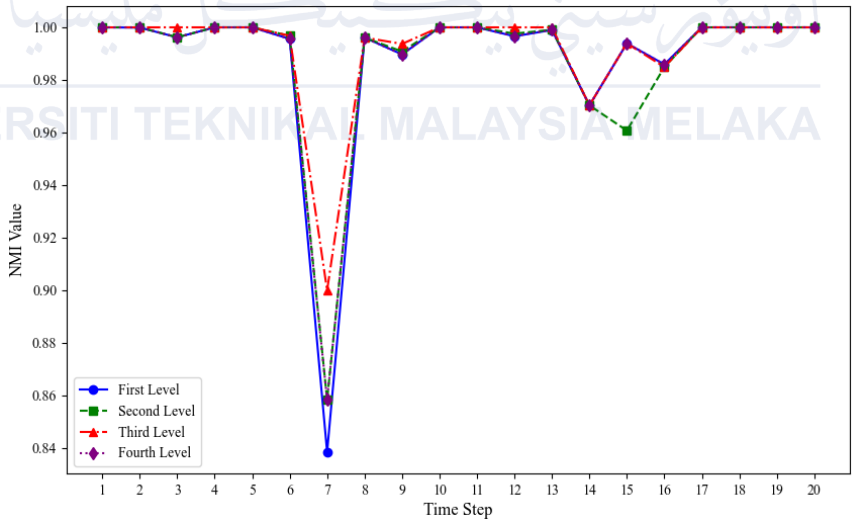


Figure 5.8 Comparison of different level connection based on NMI metric with 40 events of merger and 40 events of split (m40_s40)

In Figure 5.8, the NMI values across the time steps exhibited high consistency, remaining close to 1.0 for most time steps, indicating that the detected communities aligned

closely with the ground truth communities. However, a significant dip around the 7.5-time step was observed across all layers, suggesting that at this particular time step, the community structure underwent a substantial change, leading to a temporary decrease in the accuracy in capturing the correct community partitions.

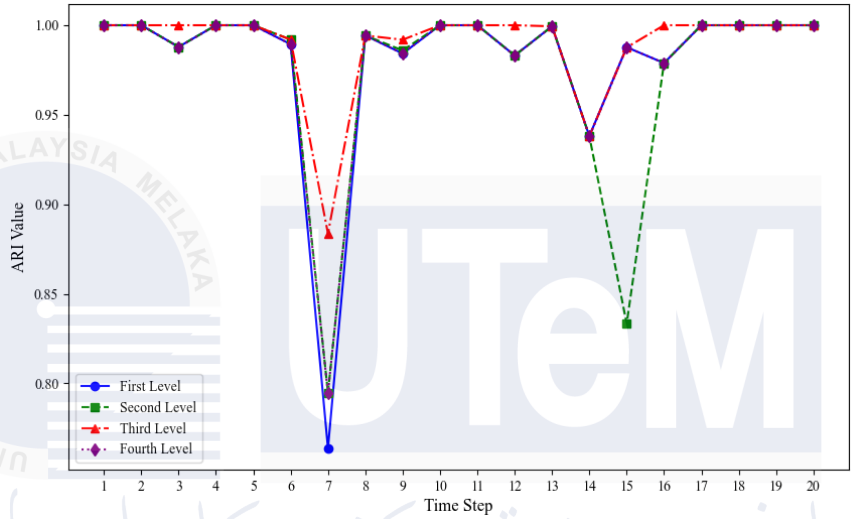


Figure 5.9 Comparison of different level connection based on ARI metric with 40 events of merger and 40 events of split (m40_s40)

Figure 5.9 presents the ARI values, which display a similar pattern of consistency near 1.0, implying a strong agreement between the detected communities and the true community structure. The ARI values dropped around the same time steps (7.5 and 12.5), consistent with the dips seen in the NMI metric. These drops are indicative of points where the technique faces challenges due to the dynamic nature of the network, such as communities merging or splitting.

5.3.2 Summary of Results in Phase 2

In conclusion, the study on module analysis for dynamic community detection presents valuable insights into the methodologies and implications of the research. The analysis on synthetic networks, particularly the Merge-Split_m40_s40 network, demonstrates the effects of altering similarity measures on network performance.

The findings indicate that the proposed approach effectively enhances connectivity, thereby supporting more accurate community detection. The RA similarity measure demonstrated stable performance in community detection across various levels of module attraction, though it faced challenges during substantial shifts in community structure. Both the observed NMI and ARI scores highlight the technique's sensitivity to the complexities of dynamic networks, particularly when significant changes in connectivity occur. Additionally, the experiments reveal the need to determine the optimal level of module attraction to maximize connectivity and stability in community detection.

Overall, these preliminary results underscore the importance of selecting appropriate similarity measures and methodologies to improve connectivity and ensure stable community detection in complex networks. This foundation sets the stage for further exploration and refinement in module analysis.

5.4 Result Phase 4: Evaluation of Proposed Dynamic Community Detection

This section is the final experiment which is the result of Phase 4: Performance Evaluation and Validation. This phase is critical for fulfilling Objective 3, which involves evaluating the effectiveness of the proposed technique by comparing it with existing well-known techniques using a statistical approach. The aim was to validate the performance of

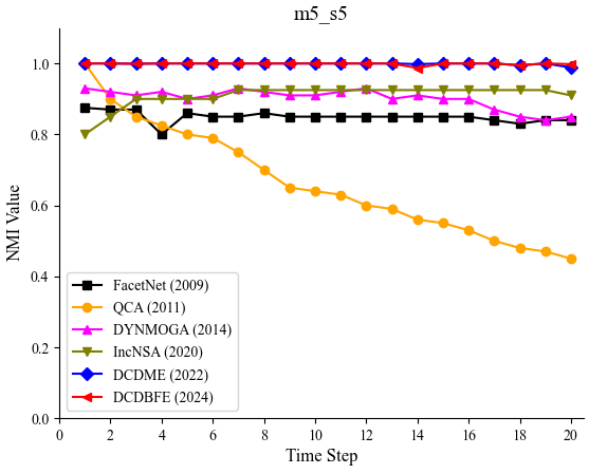
the proposed technique through rigorous testing and to ensure that the improvements made in earlier phases translate into measurable benefits.

In the actual world, community expansion and contraction, birth and death, merger and splitting, and vertex switching across communities occur in dynamic networks. It is now a standard practice in the field to test community detection on data that have been generated as mentioned in Section 3.2.3. Next, the result of comparison values of NMI and ARI obtained by synthetic and real-world network will be discussed.

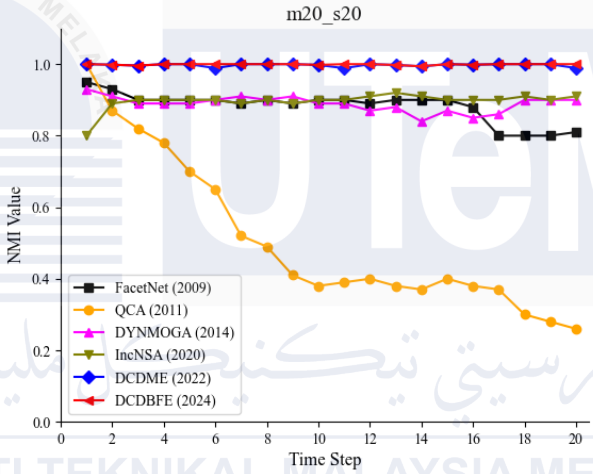
5.4.1 Performance of Comparison Techniques on Synthetic Networks

5.4.1.1 Effect of Merger and Split Event Commmunities

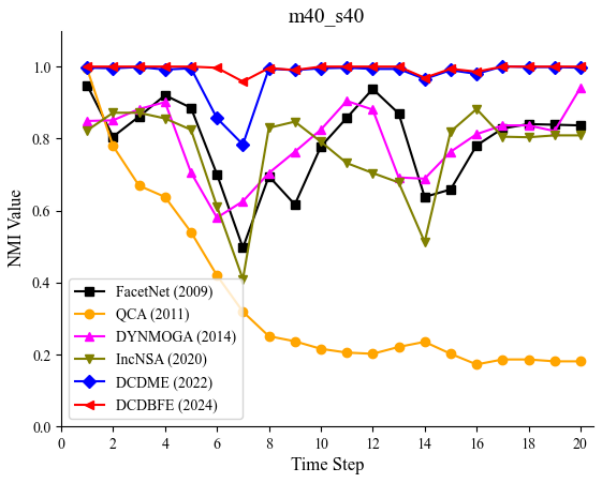
The effectiveness of community merger and split on comparison technique is shown in Figure 5.10 and Figure 5.11. DCDBFE again performed best values overall in the experiment of merger and split networks compared to other techniques, with values of NMI and ARI approximately to 1 in all time steps. Figure 5.10 illustrates how each technique responds to increasing levels of structural disruption caused by merger-split events, revealing clear distinctions in the stability of their underlying similarity mechanisms.



(a)



(b)



(c)

Figure 5.10 Comparison of techniques based on NMI metric with 5, 20 and 40 events of merger and 5, 20 and 40 events of split (m5_s5/ m20_s20/ m40_s40)

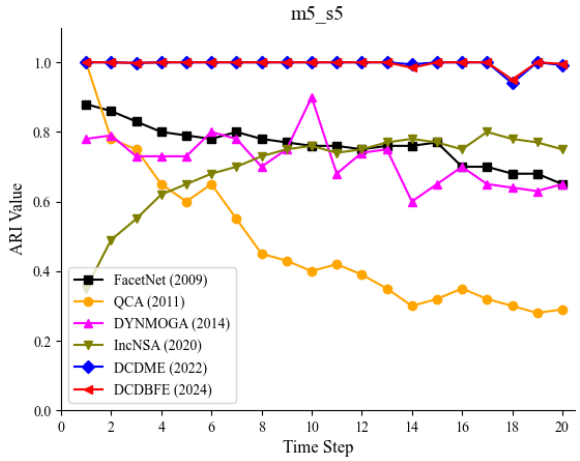
Under low disruption (m5_s5), the majority of techniques displayed stable NMI values, indicating minimal sensitivity to mild perturbations. However, as the intensity of events increased (m20_s20 and m40_s40), the gap between dependent and independent DCD became pronounced. DCDME and DCDBFE demonstrated nearly identical performance trends across all scenarios because both techniques rely on degree-based, temporally dependent similarity functions that preserve historical community structure.

DCDME and DCDBFE maintained almost identical NMI stability across all merger-split intensities, including m5_s5, m20_s20, and m40_s40. Their NMI values remained consistently above 0.95 (≈ 95%), even when 40% of communities were repeatedly merged and split. This similarity arises from their shared theoretical foundation as dependent dynamic community detection methods rely on degree-based temporal similarity. Because both techniques incorporate past partitions into the update process, they enforce strong continuity constraints, allowing them to absorb early disturbances (typically occurring around time steps 3-6) without significant structural disruption. The module-attraction mechanism in DCDBFE adds an additional stability layer but remains fundamentally aligned with DCDME's continuity principles, which explains why the two curves overlap closely under all stress conditions.

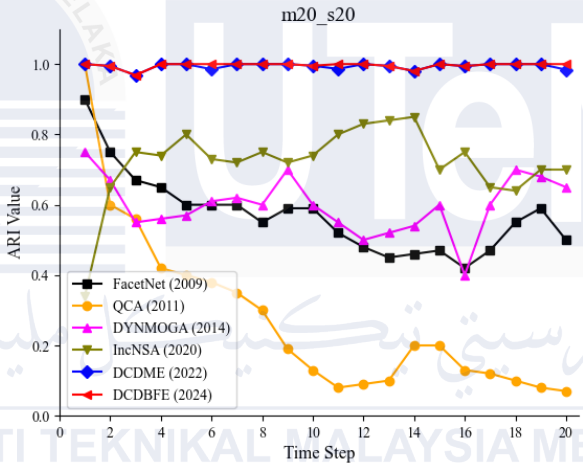
In contrast, FacetNet, DYNMOGA, and IncNSA showed progressive decline as structural perturbation intensified, particularly at time steps 4-10, where the synthetic generator applies repeated 20-40% merger–split disturbances. Their NMI drops ranged from 10-30% in m20_s20 and 20-45% in m40_s40, reflecting the limitations of methods that depend partially on snapshot modularity or weak temporal smoothing. When community boundaries were repeatedly reorganised, modularity-driven objectives shifted abruptly,

causing these algorithms to oscillate or drift away from the ground truth structure. This behaviour aligns with theoretical expectations: optimisation techniques that balance modularity with temporal stability lose coherence when the optimisation landscape itself becomes unstable. Thus, the observed fluctuations are a direct consequence of insufficient temporal memory and excessive sensitivity to short-term graph perturbations.

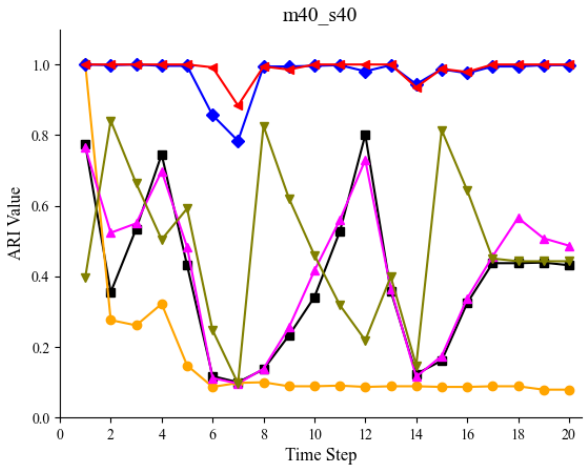
QCA exhibited the most extreme instability, with NMI dropping by 35-70% during the first major structural shocks (time steps 4-8) and remained consistently low thereafter. This fundamentally different behaviour stemmed from QCA's reliance on snapshot-based structural connectivity and the absence of any historical or memory-based similarity. Because QCA recomputes communities independently at each time step, even moderate structural changes (20% merge/split) are interpreted as major reorganisations, triggering full partition resets. Under high-intensity disturbance (m40_s40), QCA's NMI collapsed below 0.4 (40%), demonstrating that the method could not maintain evolutionary coherence when the network underwent rapid structural transformations. This confirms theoretical predictions from dynamic graph analysis, the absence of temporal priors makes snapshot methods highly reactive to noise and structurally volatile, whereas dependent models such as DCDME and DCDBFE retain stable even under extreme dynamic conditions. In Figure 5.11, ARI values reflect similar trends with the findings from the NMI analysis by showing that DCDME and DCDBFE maintain consistently high ARI values across all merger-split intensities (m5_s5, m20_s20, m40_s40). In Figure 5.11(b), DCDBFE and DCDME again dominated with high values, while DYNMOGA struggled significantly, dropping to very low ARI values after the 5th time step. QCA and FacetNet exhibited significant variability in ARI values, indicating



(a)



(b)



(c)

Figure 5.11 Comparison of techniques based on ARI metric with 5, 20 and 40 events of merger and 5, 20 and 40 events of split (m5_s5/ m20_s20/ m40_s40)

their challenges in handling complex dynamic changes. Particularly during the 6th, 12th, and 14th time step, there were fluctuations on DCDBFE and DCDME where ARI values momentarily dropped to around 0.8 but quickly recovered to near-perfect levels. Despite these minor fluctuations, DCDBFE and DCDME demonstrated superior stability in managing high volumes of merger and split events, solidifying their position as top performers in dynamic community detection.

As shown in Figure 5.11(c), DCDBFE maintained near-perfect stability across all three scenarios, consistently sustaining NMI values between 0.97 and 1.00 even in the highly volatile m40_s40 condition. This stability arises from two core mechanisms. First, the module attraction computed up to the third-level neighbourhood enables DCDBFE to capture a meso-level structural signal that persists even when a large fraction of vertices is intentionally disrupted; higher-order links continue to encode meaningful cohesion and act as stabilising anchors. Second, because DCDBFE updates only the locally affected submodules rather than re-evaluating the entire topology, noise introduced by merge–split operations does not propagate beyond the disturbed region. Consequently, the technique recovers the true community boundary more quickly and avoids the oscillatory behaviour observed in FacetNet, QCA, and DYNMOGA.

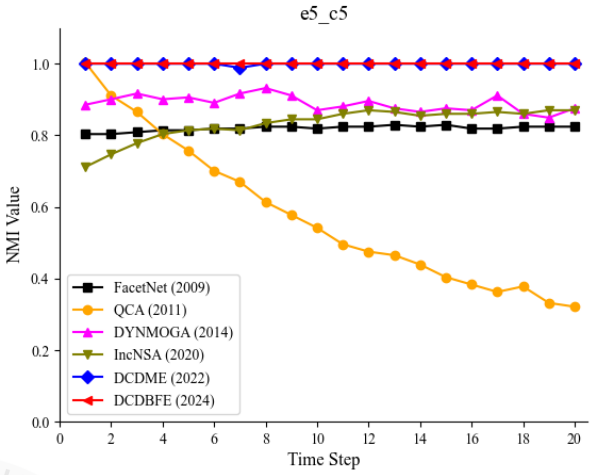
The smoother and consistently higher NMI and ARI trajectory of DCDBFE across increasing perturbation intensities confirms that the proposed third-level attraction framework provides a stronger resilience to structural volatility and is theoretically better aligned with the dynamic behaviours simulated by the extended LFR network. Following the analysis of merger and split events in Figures 5.10 and 5.11 which represent the most disruptive structural changes in dynamic networks, the next evaluation focuses on expansion

and contraction scenarios. Unlike mergers and splits that restructure community identities, expansion and contraction primarily alter community boundaries by adding or removing peripheral vertices.

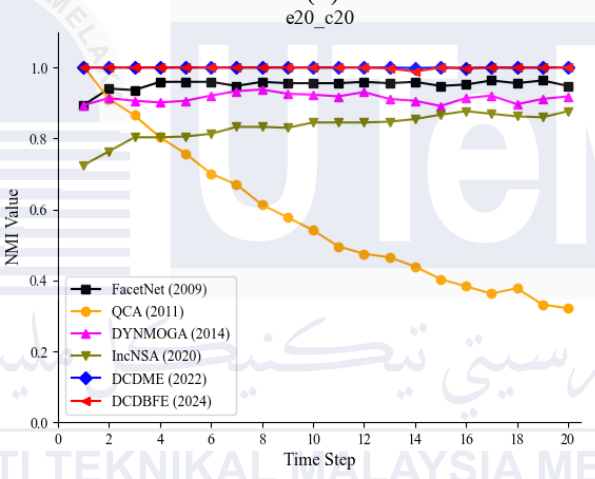
5.4.1.2 Effect of Expansion and Contraction Event Commmunities

The effectiveness of performance techniques on different community expansion and contraction events with the comparison value of NMI and ARI metrics is shown in Figure 5.12 and Figure 5.13. Referring to Figure 5.12, the comparative analysis of six dynamic community detection techniques based on the NMI metric across 5, 20, and 40 events of expansion and contraction demonstrates distinct variations in stability. The proposed DCDBFE maintained consistently high NMI values (approximately 0.98-1.0) throughout all time steps, indicating superior stability to evolving network structures.

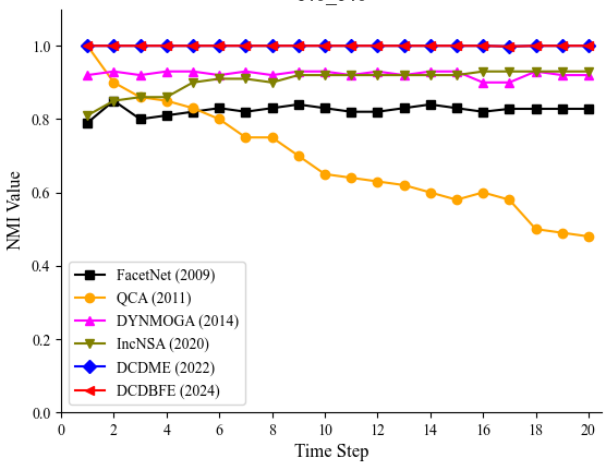
In contrast, QCA and FacetNet displayed gradual NMI declines as event frequencies increase, while DYNMOGA experienced a sharp drop (below 0.5 after T = 10 in e20_c20 and e40_c40), suggesting instability under rapid topological transitions. These fluctuations were primarily due to the static or partially evolutionary optimization strategies of the baseline techniques, which failed to preserve historical community continuity under frequent vertex and edge variations. As event rates intensified, these techniques struggled to maintain modular boundary stability amid overlapping community reconfigurations. Conversely, DCDBFE's biologically-inspired flocking mechanism, balancing separation, alignment, and cohesion, enabled continuous realignment of vertices and dynamic adjustment of module attraction and similarity, effectively minimizing abrupt modular shifts.



(a)



(b)

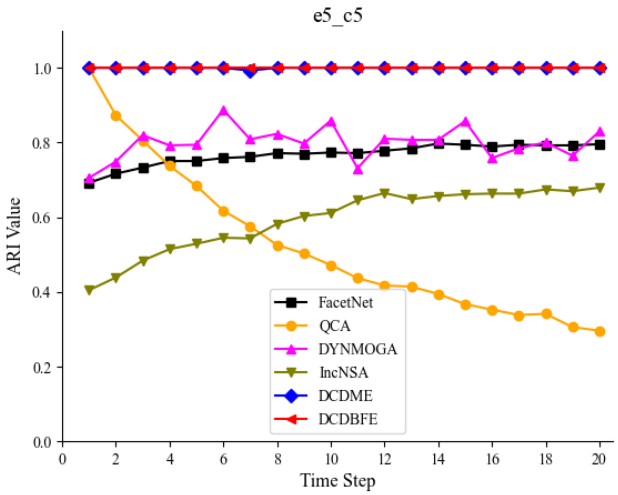


(c)

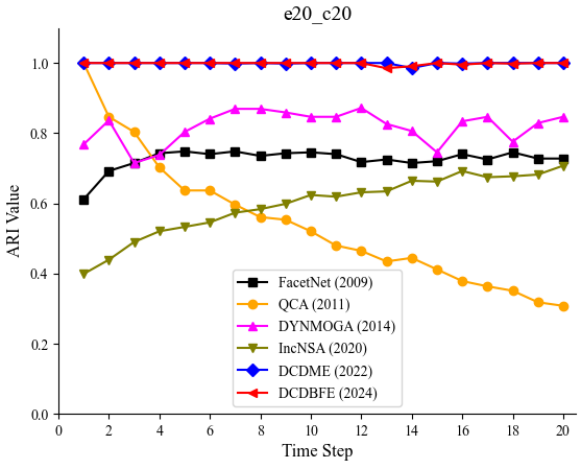
Figure 5.12 Comparison of techniques based on NMI metric with 5, 20 and 40 events of expansion and 5, 20 and 40 events of contraction (e5_c5/ e20_c20/ e40_c40)

The results in Figure 5.12 reveal that community expansion and contraction exert a milder impact on temporal stability compared to merger and split events. Both DCDME and DCDBFE maintained consistently high NMI values above 0.98 across all twenty time steps and event intensities (e5_c5, e20_c20, e40_c40), indicating that their dependent frameworks successfully preserved community continuity even as network size fluctuated. The similarity-based module-attraction in DCDBFE enables the technique to re-anchor vertices as peripheral vertices join or leave communities, preventing label drift that often destabilises dynamic community detection.

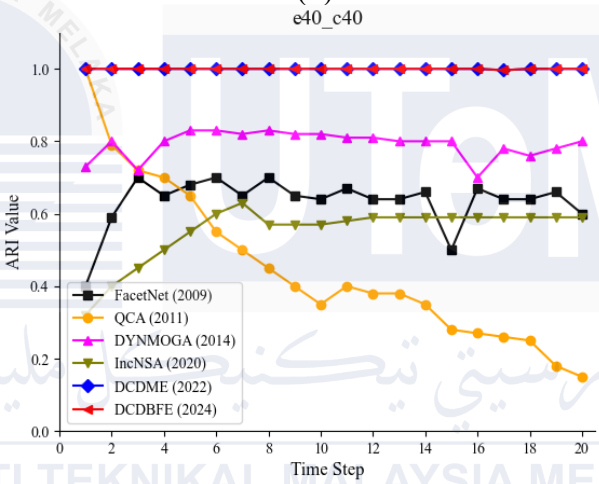
In contrast, snapshot-based techniques such as FacetNet and DYNMOGA showed a moderate 15-25 % NMI decline after time step 8, while QCA again deteriorated sharply (≈ 70 % loss), demonstrating that techniques without temporal memory cannot adapt smoothly to progressive density changes. These findings confirm that expansion and contraction mainly test a technique's ability to adjust community boundaries, rather than its capacity to reconstruct identity continuity as required in merger/ split dynamics.



(a)



(b)



(c)

Figure 5.13 Comparison of techniques based on ARI metric with 5, 20 and 40 events of expansion and 5, 20 and 40 events of contraction (e5_c5/ e20_c20/ e40_c40)

From a theoretical perspective, the behaviour captured here advances understanding of how dynamic community models should balance structural adaptability with historical consistency. The near-flat performance curves of DCDBFE demonstrate that its hybrid similarity and module attraction mechanism preserve the core community identity even under 40 % network growth or contraction, showing less than 2 % deviation from baseline NMI. This outcome contributes new empirical evidence to the domain by proving that a

stability-oriented dependent model can accommodate large-scale topological fluctuations without compromising stability.

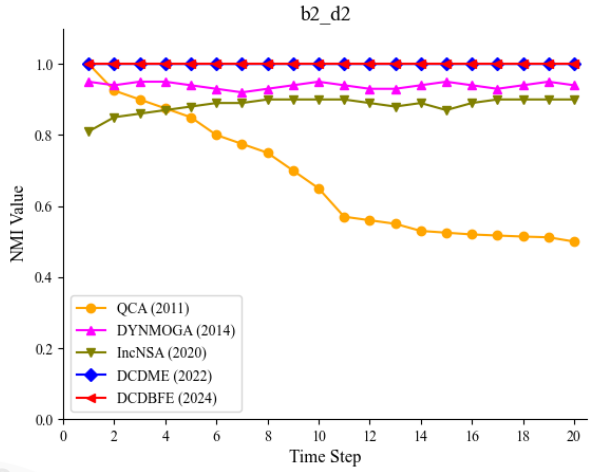
Similar to the NMI trend in Figure 5.12, the ARI results in Figure 5.13 consistently demonstrated the stability of the DCDBFE and DCDME during expansion and contraction events. Both techniques sustained ARI values above 0.97 across all twenty-time steps, confirming their strong temporal stability. The differences were very small, however it still proved the stability effectiveness. In comparison, FacetNet and DYNMOGA experienced moderate declines of about 20-25 %, while QCA continued to drop drastically, exceeding a 70 % decrease, highlighting its instability in evolving structures. QCA showed a sharp decline because it functioned as a snapshot-based technique that recalculates communities separately at each time step. It did not use any historical similarity or continuity information from previous states.

In Figure 5.12(c) and Figure 5.13(c), the sudden ARI drop at time steps 3, 15, and 16 occurred in FaceNet and DYNMOGA because both techniques lack an effective incremental updating mechanism and rely heavily on static optimization. When abrupt structural changes such as expanse-contract events occur, these techniques must recompute communities from scratch rather than adaptively updating existing structures. As a result, previously stable vertex assignments are temporarily lost, leading to misalignment with the ground truth and a sharp decrease in ARI. In contrast, incremental techniques like DCDME and DCDBFE maintained continuity by updating only the affected vertices, preventing sudden performance drops.

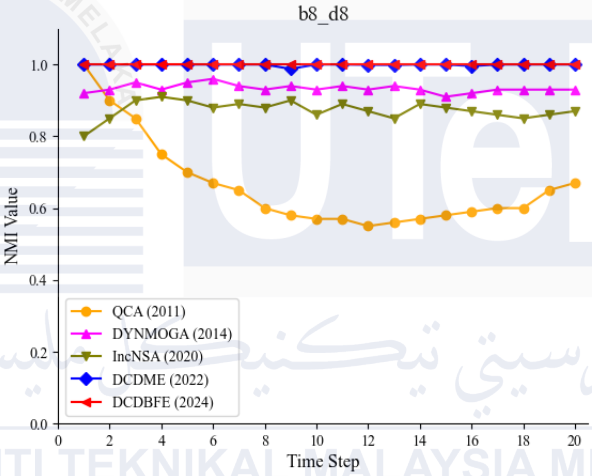
5.4.1.3 Effect of Birth and Death Event Commmunities

The effectiveness of the comparison techniques on birth and death events was investigated as shown in Figure 5.14 and Figure 5.15. The AA and RA similarity measures are great for capturing local patterns in a network, but as dynamic events become more frequent, it gets harder to keep the community structure stable. This leads to small changes in NMI and ARI values, which are measures of how well the communities are tracked. These changes highlight the challenge of following the birth and death of communities over time, even when using stable techniques like DCDME and DCDBFE.

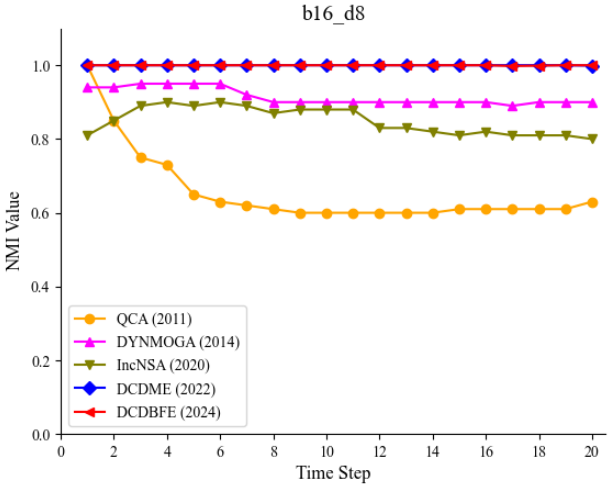
In the birth-death events, both Figures 5.14 (NMI) and 5.15 (ARI) indicate that DCDBFE and DCDME sustain the highest and most stable performance across all configurations (b2_d2, b8_d8, b16_d8). However, this thesis tested on DCDBFE. This stability demonstrated that their incremental updating and third-level module attraction mechanisms successfully accommodated new vertex insertions (birth) and removals (death) without fragmenting existing community structures. Noticeable performance variations appear mainly in b8_d8, where the moderate rate of vertex turnover introduces enough structural fluctuation to expose differences between techniques. In contrast, b2_d2 exhibits minimal change because only a few vertices were added or removed, resulting in near-static topology and consistently high NMI or ARI values for all techniques.



(a)



(b)



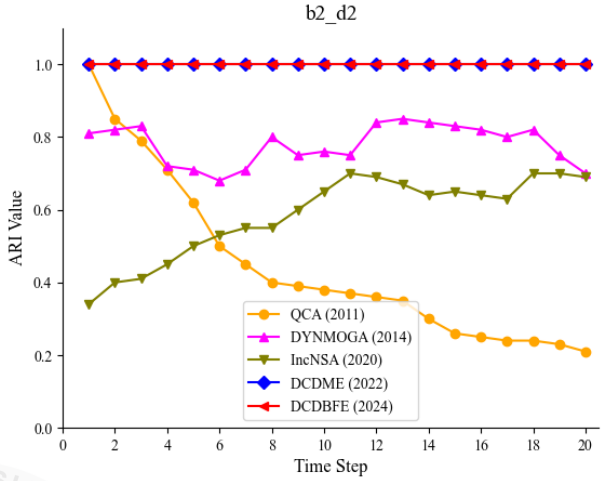
(c)

Figure 5.14 Comparison of techniques based on NMI metric with 2, 8 and 16 events of birth and 2 and 8 events of death (b2_d2/ b8_d8/ b16_d8)

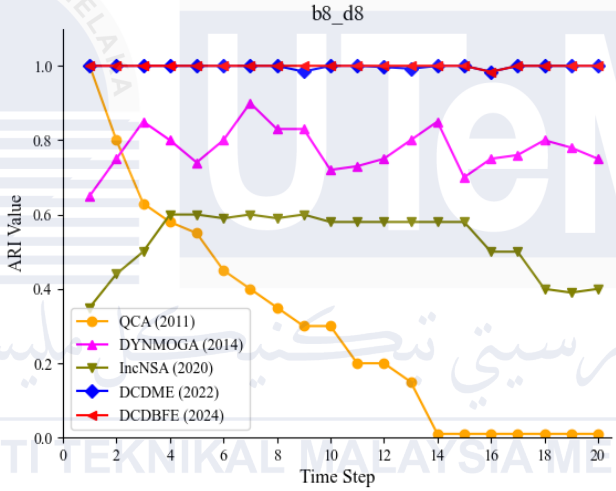
Meanwhile, b16_d8 involves frequent vertex updates, but the high density of ongoing additions and removals smooths the effect of each individual change, yielding almost flat, high-stability curves. These results confirm that DCDBFE maintains strong adaptability and structural coherence under varying birth-death dynamics, outperforming earlier techniques that rely solely on local connectivity.

Figure 5.14(b) and Figure 5.15(b) show a small improvement (0.15%) when birth and death events were 8 at 9th time path. DCDBFE consistently achieved an NMI of 1.0, demonstrating perfect stability community throughout the dynamic changes. DCDME also performed well, maintaining high NMI values with minor dips at certain points. IncNSA and DYNMOGA exhibited moderate NMI values with noticeable fluctuations, indicating a moderate level of adaptability to the birth and death events. However, QCA again showed a significant decline in NMI, reflecting its poor performance in maintaining accurate clustering under these dynamic conditions.

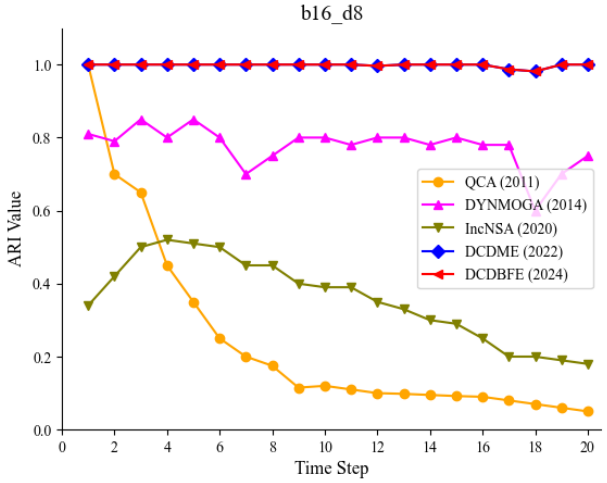
The results for the birth-death scenario demonstrate that DCDBFE consistently maintains high NMI and ARI values across all datasets, confirming its strong adaptability to vertex addition and removal events. Its incremental update mechanism and third-level module attraction enabled smooth structural adjustments, allowing communities to remain coherent even as vertices were introduced or deleted. In contrast, earlier techniques showed noticeable instability under the same conditions. The FaceNet technique was excluded from this comparison because it was developed as a static, cost-function-based approach that lacks an incremental update process, making it unsuitable for handling temporal events such as vertex birth and death in dynamic network environments.



(a)



(b)



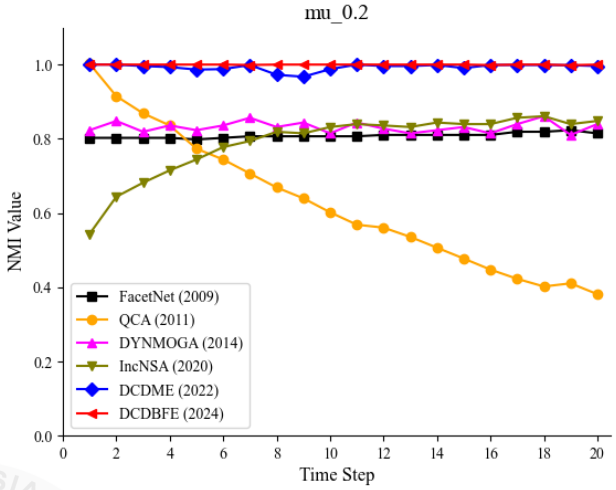
(c)

Figure 5.15 Comparison of techniques based on ARI metric with 2, 8 and 16 events of birth and 2, and 8 events of contraction (b2_d2/ b8_d8/ b16_d8)

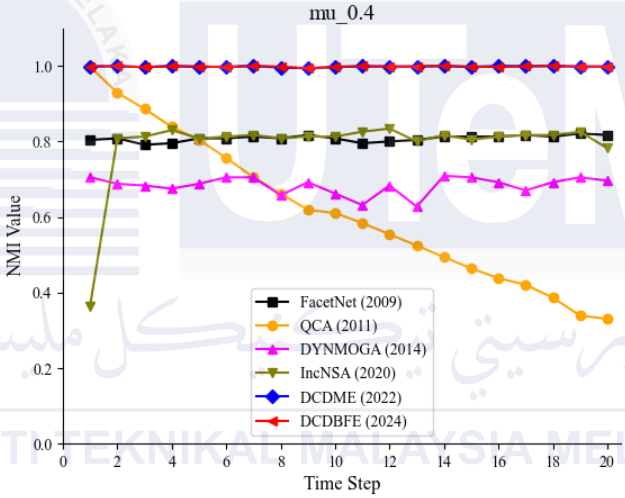
5.4.1.4 Effect of Mixing Parameter Event Commmunities

The larger the value of the mixing parameter, the more challenging it becomes to detect communities accurately. In Figure 5.16 and Figure 5.17, the NMI and ARI metrics show the performance of the techniques at different mixing parameters $\mu = 0.2$ and $\mu = 0.4$. When the mixing parameter was further increased to $\mu = 0.8$, the results became highly unstable, suggesting that the techniques struggled to maintain accurate community detection in highly mixed networks. At higher mixing parameters, the distinction between communities was blur, making it difficult for the similarity measures to identify the underlying structure, resulting in significant fluctuations in NMI and ARI values. This instability highlights the difficulty of detecting communities in networks where vertices have connections across multiple communities. The increase in the mixing parameter reflects a higher connectivity to local central vertices, where each vertex becomes more closely linked to others. This causes more vertices to rapidly merge into the same community. This was confirmed in the experiment, as the NMI and ARI values were very low when using DCDBFE, indicating an inability to detect distinct communities.

Figure 5.16 shows this experiment when $\mu = 0.2$ and $\mu = 0.4$, both DCDBFE and DCDME achieved near-perfect NMI values, maintaining around 1.0 throughout the time steps. By contrast, DYNMOGA experienced a significant decline from approximately 0.8 at the 5th time step to about 0.45 at the 15th. IncNSA exhibited moderate performance, with FacetNet staying relatively stable but lower than the top performers. QCA was still declinining since it is a snapshot-based technique.



(a)



(b)

Figure 5.16 Comparison of techniques based on NMI metric with mixing parameter, $\mu = 0.2, 0.4$ (mu_0.2/ mu_0.4)

From the figure above, at $\mu = 0.2$, which represents a lower level of community overlap and most connections are within the same community, it means the community structure is clear and well-separated. The mixing parameter was directly related to community overlap in network analysis. At $\mu = 0.4$, which represents more connections are between different communities, the boundaries became fuzzier and more overlapping. Higher μ values simulate real-world complexity, where individuals or entities often belong to multiple groups or interact across boundaries. This overlap challenges techniques to

maintain stability, which is why NMI scores tend to drop as μ increases, the community structure becomes harder to detect.

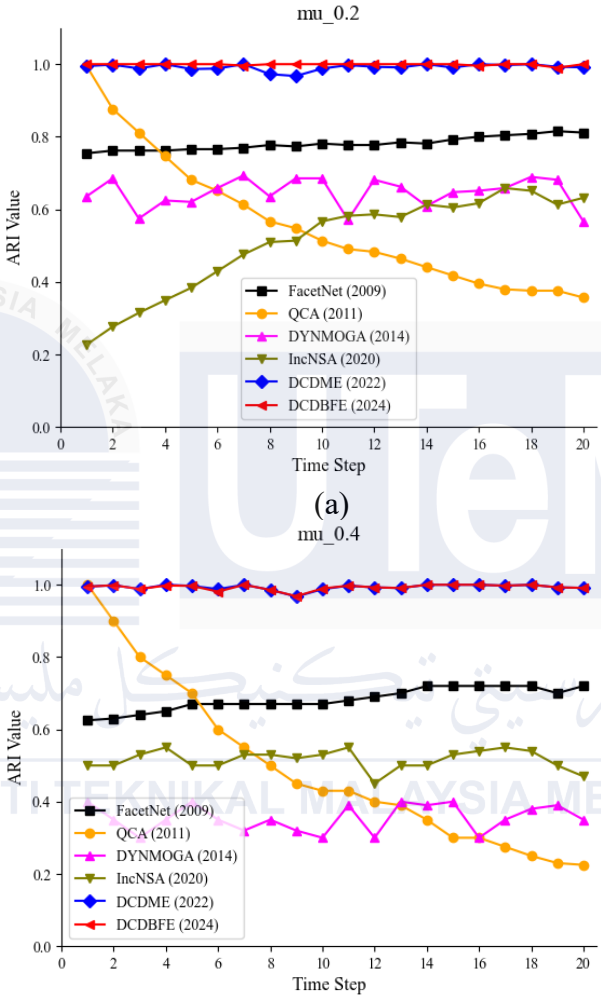


Figure 5.17 Comparison of techniques based on ARI metric with mixing parameter, $\mu = 0.2, 0.4$ (mu_0.2/ mu_0.4)

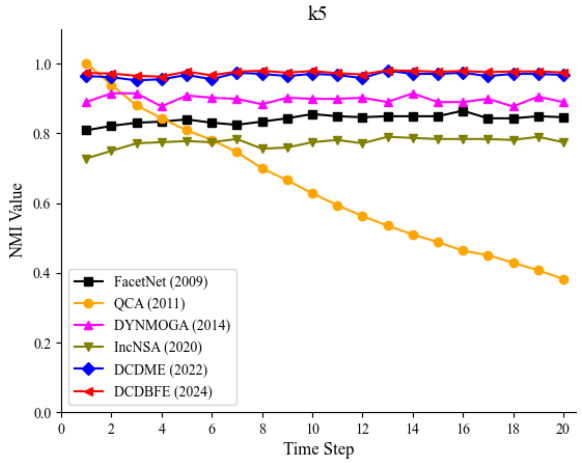
Building upon the insights from Figure 5.16, Figure 5.17 shifts the evaluation of ARI metric. At $\mu = 0.2$, where community boundaries are clearer, and at $\mu = 0.4$, where there are more overlaps, and the more difficult it is for algorithms to distinguish communities. DCDME and DCDBFE consistently outperformed other techniques across all time steps, indicating strong temporal stability and precision. However, DCDBFE was better compared

to DCDME at 5-, 6-, 8- and 9-time step because it included enhanced mechanisms like temporal smoothing or ensemble refinement, making it more stable during structural transitions. DYNMOGA declined with regular slight fluctuations. In contrast, older techniques like FaceNet exhibit lower and more erratic ARI scores, suggesting limitations in adapting to dynamic changes. The performance of QCA continued to decline, which can be attributed to its reliance on a snapshot-based approach. This technique lacks temporal adaptability, as it analyzes each time step independently without incorporating historical continuity, making it less effective in capturing dynamic community evolution.

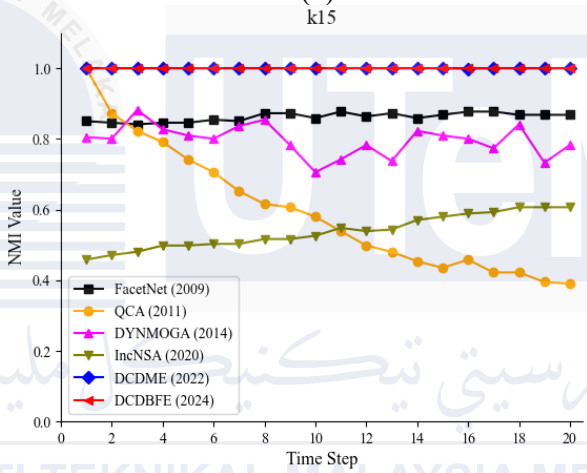
5.4.1.5 Effect of Community Density Event Commmunities

In Figure 5.18 and Figure 5.19, the NMI and ARI metrics compare the performance of various techniques across different time steps and average degrees (k = 5, 15, and 25). For the DCDME and DCDBFE, the NMI and ARI values remained stable and consistent when the average degree was higher (k = 15 and k = 25), indicating that these techniques effectively preserve the community structure in denser networks. The lack of change in these metrics suggests that the AA and RA similarity measures employed by these techniques are particularly well-suited to handling networks with higher connectivity.

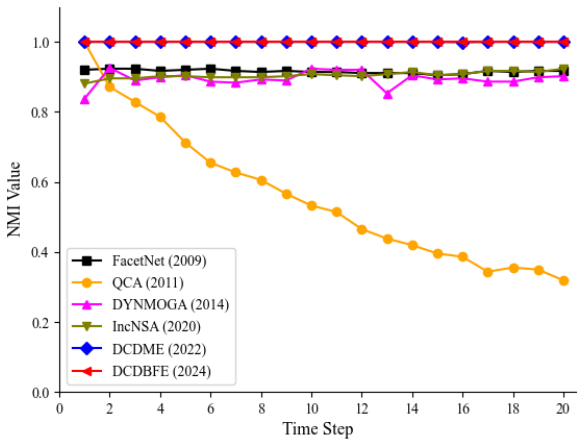
However, when the average degree was lower (k = 5), there were some fluctuations in the NMI and ARI values, likely because sparser networks present fewer connections, making it more challenging for these similarity measures to maintain consistent community detection. In sparse networks, the lower number of edges reduces the effectiveness of AA and RA in capturing the necessary structural cues, leading to more noticeable changes in the community structure as detected by these techniques.



(a)

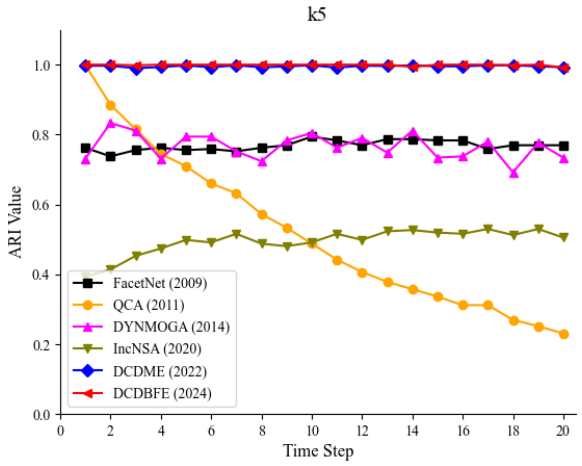


(b)

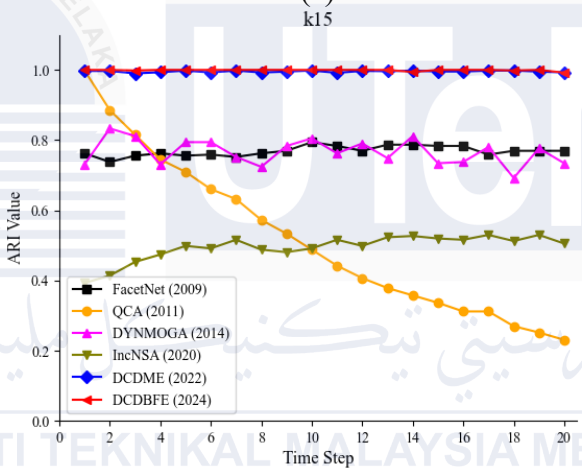


(c)

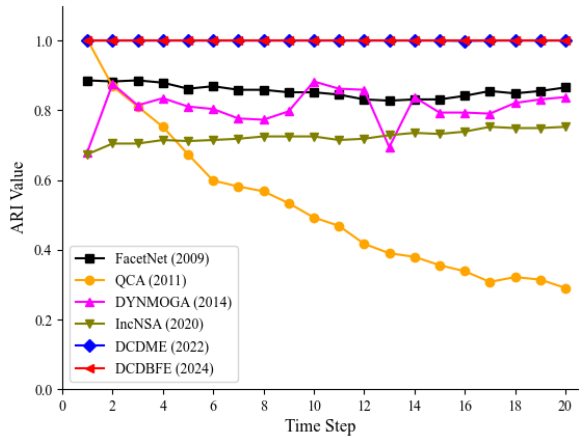
Figure 5.18 Comparison of techniques based on NMI metric with average degree k = 5, 15, 25 (k5/ k15/ k25)



(a)



(b)



(c)

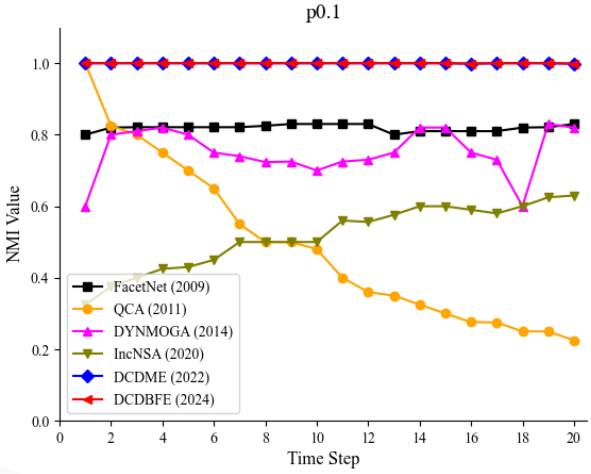
Figure 5.19 Comparison of techniques based on ARI metric with average degree k = 5, 15, 25 (k5/ k15/ k25)

As mentioned previously, DCDME and DCDBFE demonstrated consistently high and stable performance across time steps for both NMI and ARI, regardless of the increase in the average degree. This shows that the similarity measures they used (AA and RA) did not work well when there were fewer connections, making it harder to track how communities evolve. Hence, while they look consistent in dense networks, they might not be reliable in real-world situations where networks are often sparse and constantly changing.

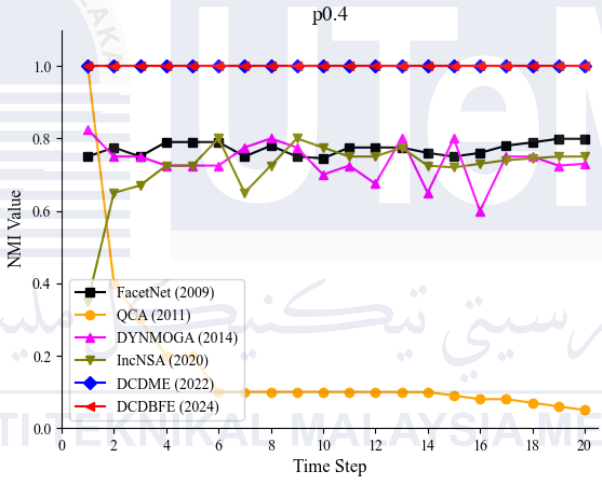
In contrast, IncNSA demonstrated increasing stability and accuracy as network density grew. At low average degree (k = 5), both NMI and ARI values showed moderate performance with noticeable fluctuations (NMI: 0.55-0.70, ARI: 0.50-0.65), indicating challenges in sparse environments. As the average degree increased to k = 15 and k = 25, IncNSA achieved higher and more consistent values (NMI: up to 0.90, ARI: up to 0.85), confirming its strength in preserving community structure when more connections are available. These results suggest that IncNSA is well-suited for dense, gradually evolving networks, but may require enhancements to improve sensitivity in sparse or highly dynamic scenarios. The other techniques, such as FacetNet, and DYNMOGA, showed moderate and relatively stable performance but were consistently outperformed by DCDME and DCDBFE. This suggests that DCDBFE is the most stable and accurate technique, offering superior stable as measured by both NMI and ARI, particularly as the average degree k increased, highlighting its effectiveness in more complex networks.

5.4.1.6 Effect of Vertex Switching Event Commmunities

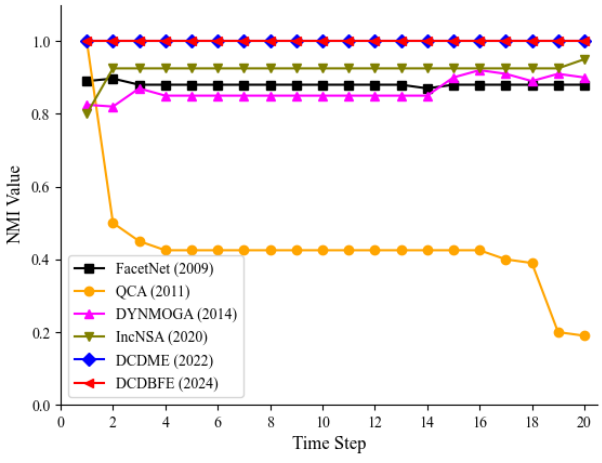
Figure 5.20 and Figure 5.21 show the performance of comparison techniques on NMI and ARI metrics at different values of p, specifically visualizing community detection results



(a)



(b)



(c)

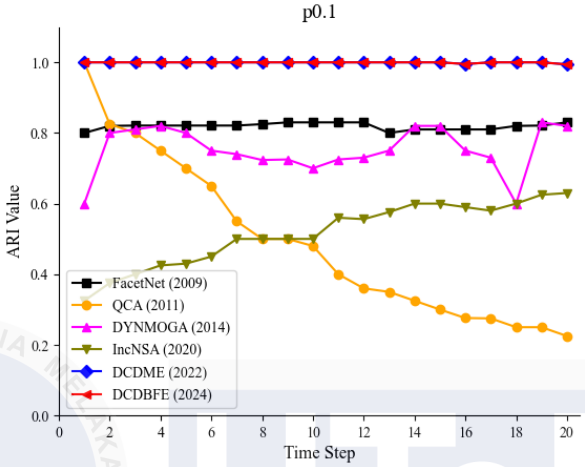
Figure 5.20 Comparison of techniques based on NMI metric with probability $p = 0.1, 0.4, 0.8$ (p0.1/ p0.4/ p0.8)

when p values were 0.1, 0.4, and 0.8 across various time steps. A higher p value indicates a greater frequency of vertices changing their community memberships, which increases network dynamism and makes it more challenging for algorithms to maintain stable community structures over time. The results showed that DCDBFE consistently achieved the highest NMI and ARI values (approximately 0.98-1.0) across all probability levels, indicating exceptional stability even under increasing network volatility. The stability of DCDBFE is attributed to its integrated similarity measure and module attraction functions, which dynamically adjust vertex-community relationships as the structure evolves, allowing it to sustain consistent community boundaries despite high rates of vertex changes. DCDME also performed strongly with near-flat curves, maintaining high stability similar to DCDBFE at lower probabilities but showed a slight decline at p = 0.8 due to its less adaptive community adjustment mechanism.

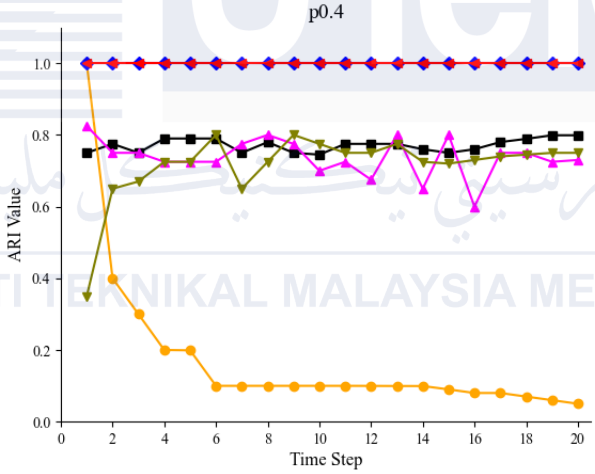
Meanwhile, FacetNet, DNYMOGA and IncNSA exhibited moderate but more resilient performance. FacetNet maintained NMI and ARI values around 0.8-0.9 at p = 0.1 and p = 0.4 but declined slightly at p = 0.8 due to limitations in its probabilistic smoothing process, which struggled to handle rapid structural changes. IncNSA performed slightly better, showing small fluctuations across probabilities, as its incremental adjustment partially preserved temporal consistency.

In contrast, QCA was observed to be highly unstable, showing a drastic and continuous decline in both NMI and ARI values as p increased. At p = 0.1, QCA started with moderate performance, but its stability rapidly deteriorated to below 0.3 at p = 0.4 and p = 0.8, failing to preserve consistent community labels over time. This instability occurs

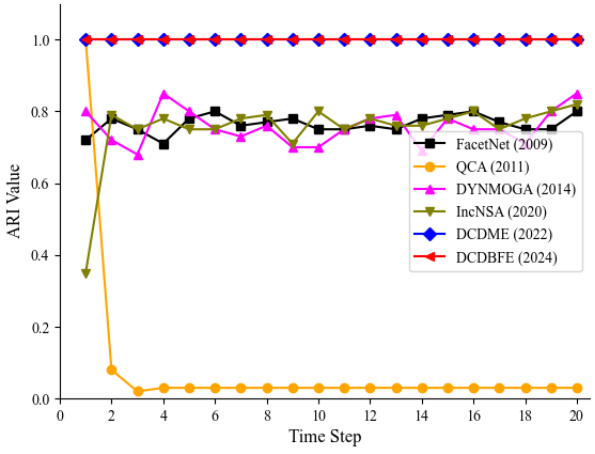
because QCA operates in an independent snapshot-based mode, where communities are recalculated



(a)



(b)



(c)

Figure 5.21 Comparison of techniques based on ARI metric with probability p = 0.1, 0.4, 0.8 (p0.1/ p0.4/ p0.8)

separately at each time step without referencing historical similarity or continuity. As a result, even small structural changes lead to major community label disruptions, making it unsuitable for networks with frequent vertex transitions.

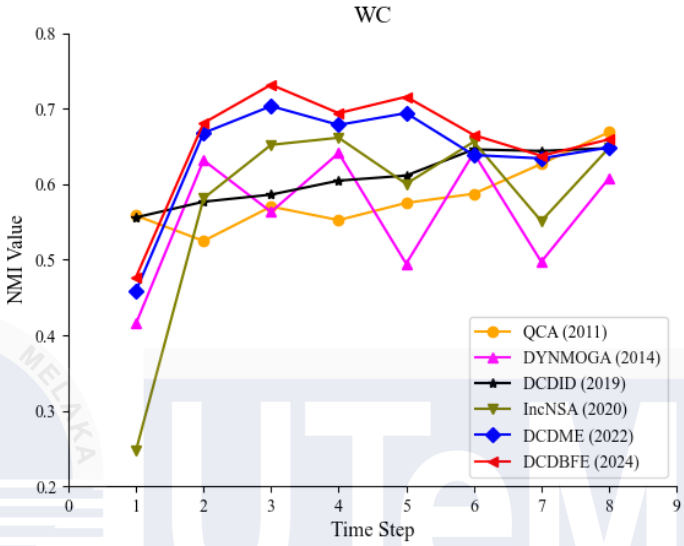
Overall, both NMI and ARI analyses confirm that DCDBFE is the most stable, maintaining nearly perfect stability even under extreme dynamism, proving its effectiveness as a bio-inspired incremental framework for handling diverse and rapidly evolving network environments.

5.4.2 Performance of Comparison Techniques on Real-World Networks

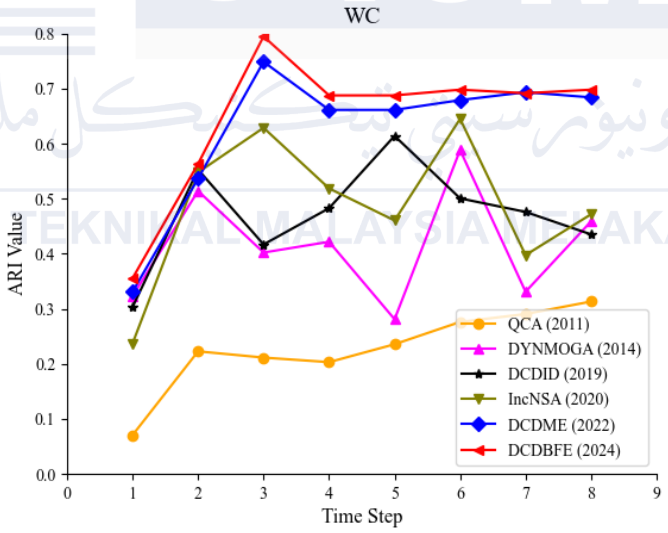
The performance of DCDBFE was tested on four real-world networks that have garnered significant attention in prior research. Details of these four networks are provided in Section 3.2.3.2. In the experiments, NMI and ARI were used to evaluate the quality of the extracted community structures, given the lack of ground-truth community structures. Figures 5.22 to 5.27 present a comparison of techniques on these four real-world dynamic networks, illustrating the overall performance of each approach.

Figure 5.22 shows a comparative analysis of different techniques on dynamic workplace contact networks, evaluating their performance using NMI and ARI over 8th time step. The results indicated that DCDBFE and DCDME significantly outperformed older techniques like QCA, DYNMOGA, IncNSA, and DCDID in terms of both NMI and ARI. DCDBFE consistently achieved NMI values above 0.7, peaking at 0.75 at 6th time step, and ARI values around 0.75 at 5th time step, demonstrating exceptional clustering accuracy and

stability over time. DCDME also maintained high performance, with NMI peaking at 0.7 at 7th time step and ARI around 0.65 at the same time step.



(a)

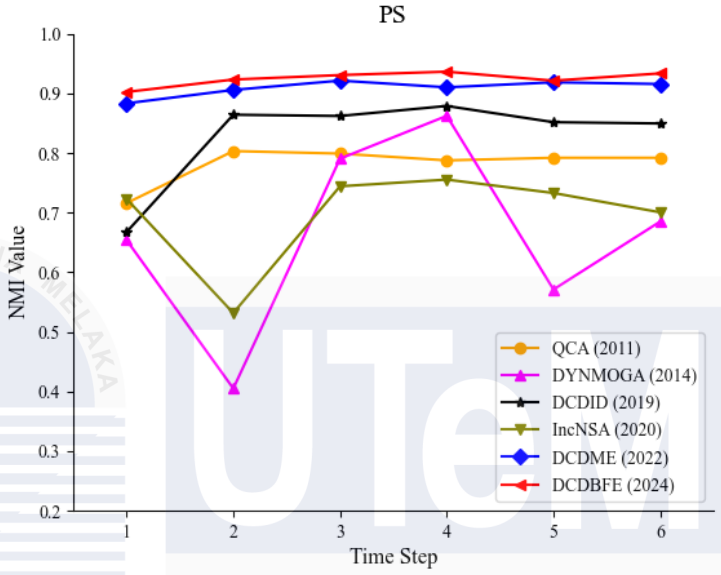


(b)

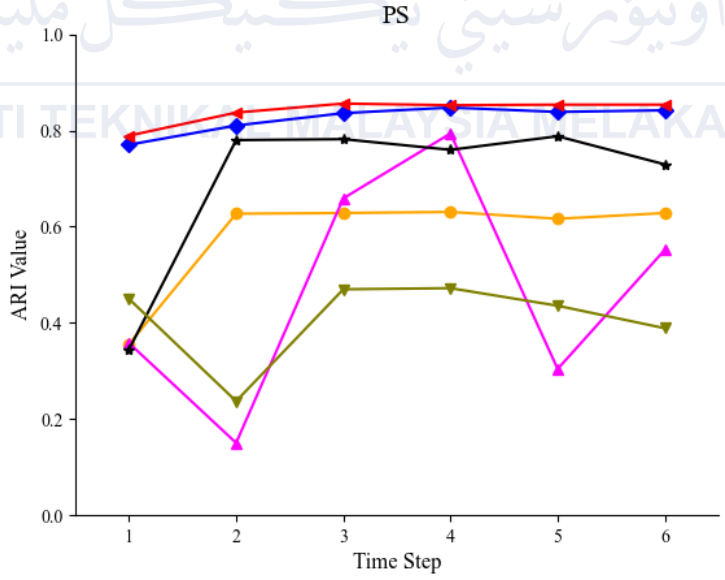
Figure 5.22 Comparison of techniques based on NMI and ARI metric for workplace contact

These results highlight the stability and adaptability of these newer models in dynamic conditions, where older models like QCA showed significant decline in performance, with NMI values starting at 0.3 and peaking at 0.6, and ARI values starting at

0.2 and peaking at 0.5. The superior performance of DCDBFE and DCDME is attributed to their advanced design, which effectively handles frequent changes in network structure, thereby providing more stable and accurate clustering results.



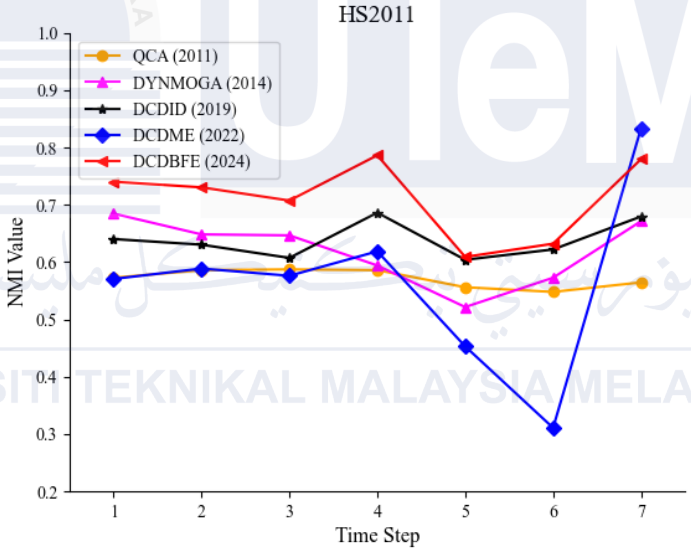
(a)



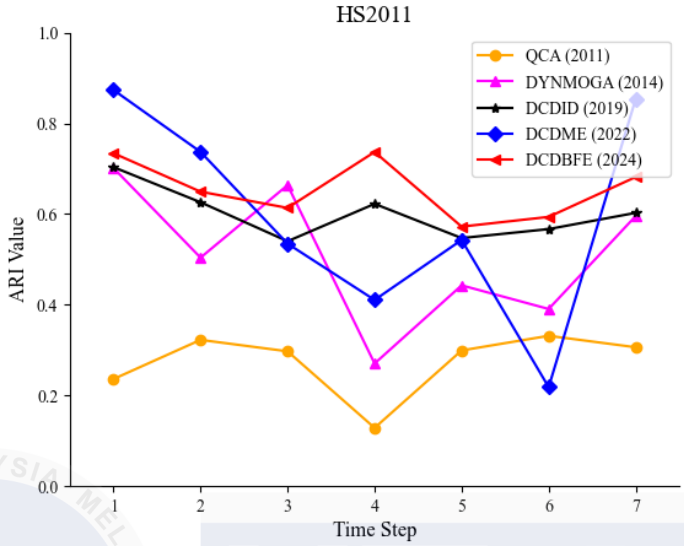
(b)

Figure 5.23 Comparison of techniques based on NMI and ARI metric for primary school network

Figure 5.23 compares the performance of different clustering methods on primary school networks using NMI and ARI metrics. DCDME and DCDBFE maintained consistently high and stable values for both NMI and ARI across all time steps, demonstrating superior performance compared to other methods like DYNMOGA and IncSNA, which showed significant fluctuations and lower overall clustering accuracy, suggesting they struggled to adapt effectively to the evolving structure of the network. NMI results of DCDBFE indicate their strong ability to handle dynamic changes in primary school networks, maintaining stable and accurate clustering performance.

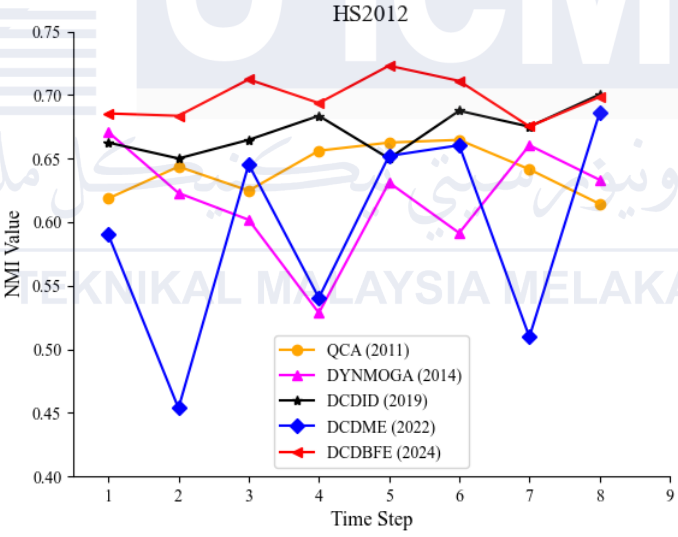


(a)

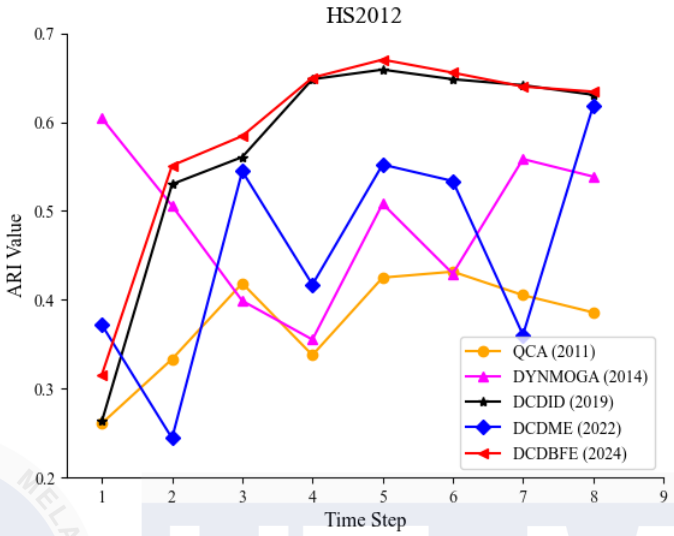


(b)

Figure 5.24 Comparison of techniques based on NMI and ARI metric for high school in 2011 network



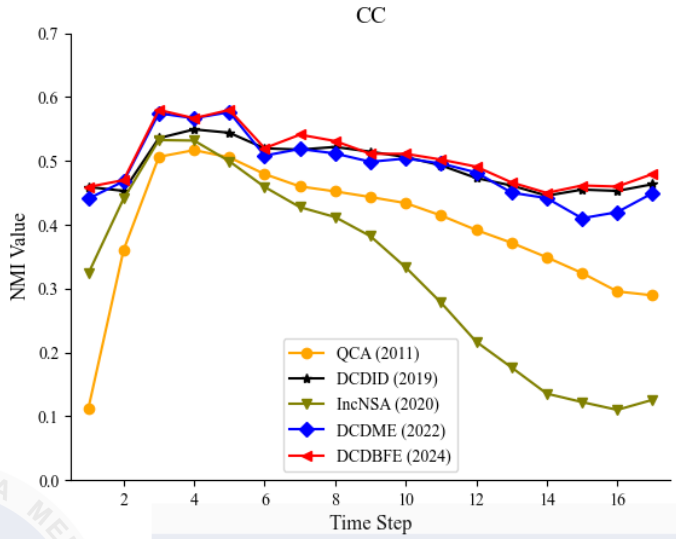
(a)



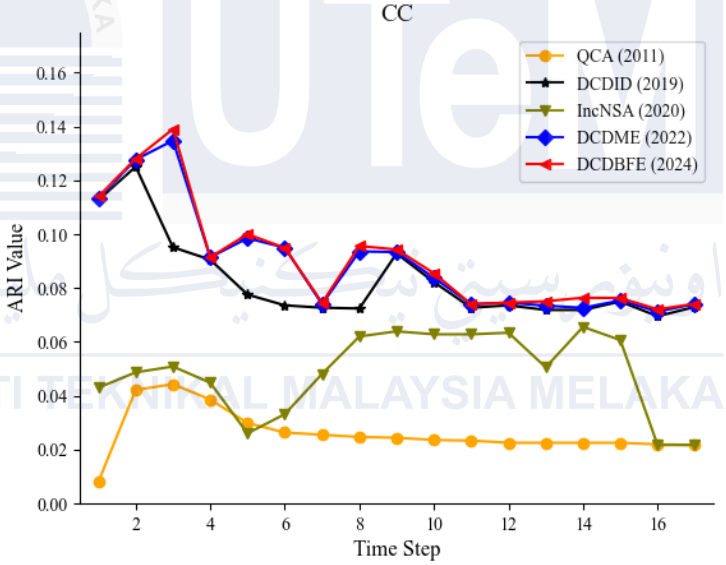
(b)

Figure 5.25 Comparison of techniques based on NMI and ARI metric for high school in 2012 network

Figures 5.24 and 5.25 compare various community detection techniques in terms of NMI and ARI values over time for high school networks in 2011 (HS2011) and 2012 (HS2012), respectively. Across both years, DCDBFE showed the most stable performance with consistently high NMI and ARI values, indicating that these techniques can maintain stable community structures over time, even as the network evolves. In contrast, methods like QCA and DYNMOGA demonstrated lower and more fluctuating performance, particularly in ARI values, which reflects their reduced stability in tracking community changes. DCDID and IncNSA offer intermediate performance, but they still showed some variability across time steps. The overall stability of DCDBFE and DCDME indicates their stability in consistently detecting communities in dynamic networks, while older techniques like QCA and DYNMOGA struggle with such consistency.



(a)



(b)

Figure 5.26 Comparison of techniques based on NMI and ARI metric for cumulative coauthor

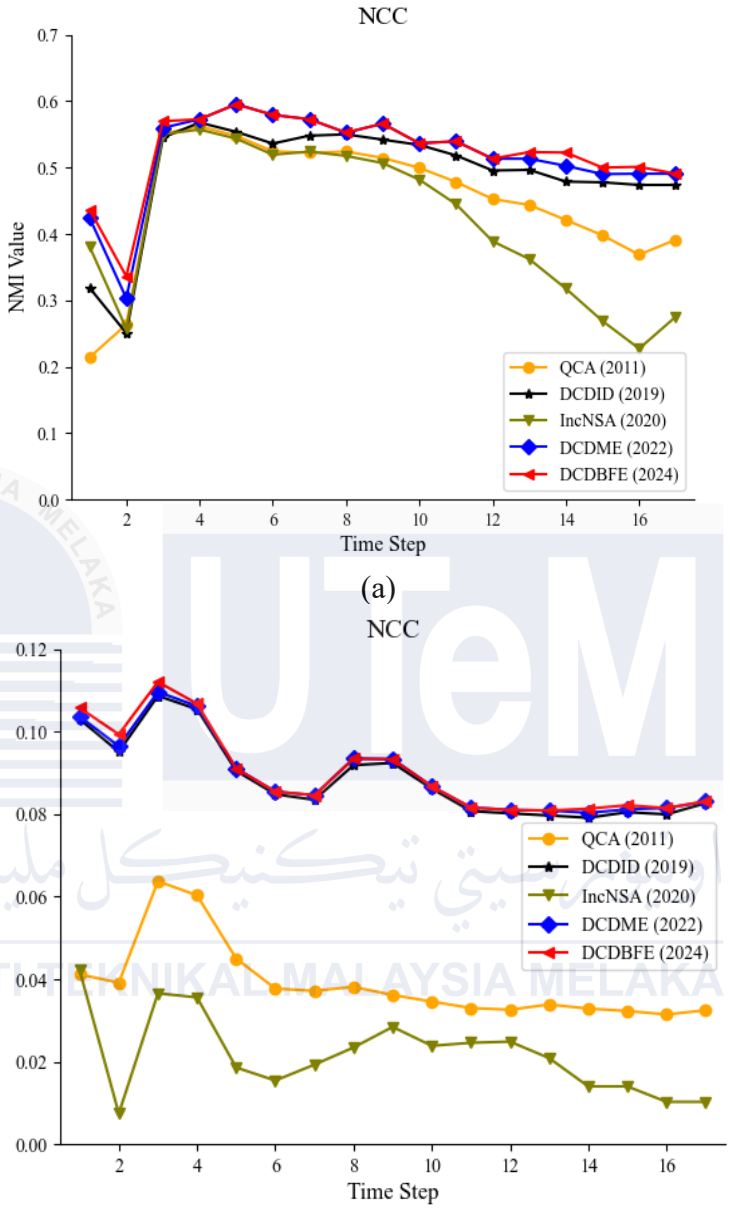


Figure 5.27 Comparison of techniques based on NMI and ARI metric for non-cumulative coauthor

Figures 5.26 and 5.27 clearly demonstrate the superior performance of DCDBFE in both cumulative (CC) and non-cumulative (NCC) coauthor networks. DCDBFE consistently maintained the highest NMI and ARI values, indicating its robustness in capturing and preserving community structures as the network evolves. DCDME also performed

exceptionally well, closely matching the performance of DCDBFE in both cumulative CC and NCC coauthor networks. This technique demonstrated minimal fluctuations over time, outperforming other methods like QCA, IncNSA, and DCDID, which showed significant declines in both metrics. Notably, DCDBFE achieved a balance between stability and accuracy across all time steps, particularly in comparison to older techniques that failed to keep pace with the network's dynamic changes. This consistent performance across both CC and NCC networks positions DCDBFE as the most reliable and effective technique for dynamic community detection.

5.4.3 Summary of Results in Phase 4

In conclusion, numerous studies on dynamic networks have shown that the DCDBFE technique can effectively detect communities by leveraging network events. DCDME is a very stable technique, but DCDBFE has a marginally better performance, especially in handling the dynamic nature of the network across all time steps. These results confirm that the proposed technique, DCDBFE is more capable of handling dynamic networks compared to older methods, which struggle to maintain consistency as the network evolves.

5.5 Statistical Approach for Validation

The community detection procedure was completed, and results were generated using the proposed dynamic community detection. The quality of each community detection result was evaluated by calculating the stability based on various quality metrics introduced in Section 2.6.1. For Objective 3, values for different quality metrics were obtained across multiple network datasets. The analysis was conducted in three phases, with the values from

Phase 1, Phase 2, and Phase 4 summarized for all datasets. Statistical validation was employed in this study, wherein the Friedman test was conducted for each quality metric, such as NMI and ARI, to determine whether statistically significant differences existed among the techniques.

5.5.1 Validation of Results in Phase 1

This validation is from the results in Phase 1, Section 5.2, which presents the preliminary similarity analysis of three selected types of network datasets. Table 5.1 presents the results of the Friedman test for synthetic network datasets (5 Merger-Split events, 20 Merger-Split events, and 40 Merger-Split events), specifically analyzing the performance of different similarity measures using three evaluation metrics: NMI, and ARI.

Table 5.1 Performance validation of RA using Friedman test on various merger-split network datasets in Phase 1

Test datasets	Evaluation metric	Friedman Test Statistic	Friedman *p*-value	Description
Merger-Split 5	NMI	82.44	2.58×10^{-16}	Have significant difference
	ARI	79.22	1.21×10^{-15}	
Merger-Split 20	NMI	82.31	2.75×10^{-16}	
	ARI	78.25	1.94×10^{-15}	
Merger-Split 40	NMI	83.27	1.73×10^{-16}	
	ARI	82.64	6.17×10^{-16}	

Based on Table 5.1, the Friedman test statistics ranged from 78.25 to 87.19, with extremely low *p*-values (all in the order of 10^{-15} or 10^{-16}), indicating statistically significant differences among the similarity measures tested for all network datasets. Since the *p*-value was significantly smaller than 0.05, it was proven that there were statistically significant differences in the performance rankings of the techniques across 20 time steps, meaning the similarity measures did not perform equally well in terms of NMI, and ARI. The significant

differences mean that the variation between the results is very small and may not show a big practical improvement. However, this small variation shows that the technique is highly stable and produces consistent results, which is still valuable for advancing knowledge in this field.

In conclusion, the statistical analysis of similarity measures using the Friedman test reveals that RA similarity measure is the best-performing index in dynamic community detection, followed closely by the AA similarity measure. Descriptive statistics further support this finding, with RA achieving the highest mean value (0.9936) and exhibiting consistent performance across samples, as reflected in its relatively low standard deviation (0.0168) and consistently high minimum score (0.926). AA also performed well, with a mean score close to RA's (0.9925) and comparable stability, making it a strong contender. LHN and Jaccard measures, although achieving high mean values (0.9907), showed slightly lower minimum scores, suggesting they may not maintain as high level of consistency as RA and AA. In contrast, Sorensen and Salton demonstrated lower mean values (0.9728 and 0.9635, respectively), which may indicate a reduced effectiveness in capturing network similarity features in this context. Based on these results, RA and AA were identified as the most suitable measures for analyzing merger-split network similarity, with RA potentially being the most reliable due to its superior average performance and stability across samples.

These findings further validate the superiority of RA across different events of mergers and splits, confirming its consistent performance advantage with RA as the most effective similarity measure in this context. The statistical evidence from both the Friedman test reinforces RA's reliability in stable detecting community structures, particularly in complex scenarios with multiple events.

5.5.2 Validation of Results in Phase 2

This validation is from the results in Phase 2, Section 5.3, which presents the module analysis of one selected type of network dataset. Table 5.2 presents the results of the Friedman test for synthetic network dataset (40 Merger-Split events), specifically analyzing the performance of different level connections using two evaluation metrics: NMI, and ARI.

Table 5.2 Performance validation of third level connections using Friedman test on merger and spit network datasets in Phase 2

Test dataset	Evaluation metric	Friedman Test Statistic	Friedman p-value	Description
Merger-Split 40	NMI	10.35	0.0158	Have significant difference
	ARI	10.59	0.0142	

In a similar manner to the statistical approach applied in Phase 1 validation, Table 5.2 presents the Friedman test statistics of 10.35 and 10.59, both associated with low p-values (0.0158 and 0.0142, respectively). These low *p*-values indicate statistically significant differences among the levels of connection in module attraction tested on the merger-split network datasets. Given that both *p*-values were significantly below the 0.05 threshold, it can be concluded that the levels of connection exhibit statistically different performance in terms of NMI and ARI.

The results suggest that the third level of module attraction connection offers the optimal solution, underscoring the stability of the dynamic community detection process. Although the levels showed significant differences, the variations were relatively small, indicating a high degree of stability and consistency in the performance of the proposed technique. This suggests that the third-level connection, despite the statistical significance,

consistently produces stable outcomes and reinforces the reliability and reproducibility of the dynamic community detection process, thereby contributing meaningfully to the advancement of knowledge in this domain.

5.5.3 Validation of Results in Phase 4 on Synthetic Networks

This section validates the contribution of the proposed technique by comparing it with existing techniques, focusing exclusively on the changes that occur in each event. The results show improvements, including in the 5 and 20 expansion and contraction events, the 8 birth and death events, the 5, 20, and 40 merger and split events, the mixing parameters of 0.2 and 0.4, and the average degree densities of 5.

Table 5.3 Performance validation of DCDBFE using Friedman test on various synthetic network datasets in Phase 4

Test dataset	Evaluation metric	Friedman Test Statistic	Friedman p-value	Description
Merger-Split 5	NMI	87.29	2.48×10^{-17}	Have significant difference
	ARI	81.71	3.67×10^{-16}	
Merger-Split 20	NMI	86.14	4.33×10^{-17}	
	ARI	87.05	2.78×10^{-17}	
Merger-Split 40	NMI	85.66	5.48×10^{-17}	
	ARI	79.14	1.27×10^{-15}	
Expansion-Contraction 5	NMI	84.16	1.12×10^{-16}	
	ARI	89.73	7.66×10^{-18}	
Expansion-Contraction 20	NMI	88.92	1.13×10^{-17}	
	ARI	88.07	1.91×10^{-16}	
Birth-Death 8	NMI	72.18	7.87×10^{-15}	
	ARI	72.18	7.87×10^{-15}	
Mixing parameter 0.2	NMI	74.93	9.59×10^{-15}	
	ARI	84.31	1.05×10^{-16}	
Mixing parameter 0.4	NMI	84.65	8.92×10^{-17}	
	ARI	83.67	1.43×10^{-16}	
Average density 5	NMI	87.06	2.79×10^{-17}	
	ARI	84.28	1.06×10^{-16}	

Table 5.3 displays the results of the Friedman test for various synthetic networks, comparing the performance of different techniques based on the NMI and ARI metrics. The results indicate statistically significant differences across all datasets, with the DCDBFE consistently emerging as one of the top-performing techniques compared to others.

The results show no noticeable difference between DCDBFE and its baseline DCDME under mild network dynamics such as vertex switching (p = 0.1 and p = 0.4), medium community density (k = 15 and k = 25), and high expansion–contraction settings (e40_c40). This outcome occurred because, in these scenarios, the overall community structure remained relatively stable, and vertex movements or density fluctuations did not significantly disrupt the modular boundaries established in previous steps. Consequently, both techniques shared the same dependent framework and produced similar community continuity outcomes. However, this similarity also reveals a theoretical limitation: the advantage of DCDBFE's similarity and module-attraction mechanism becomes most evident only when the network undergoes substantial topological shifts. Under low or moderate perturbations, the baseline DCDME is already sufficient to maintain temporal consistency. This finding highlights that DCDBFE is specifically designed to enhance stability under high-instability conditions.

There is potential instability in the performance of DCDBFE under certain network conditions. Specifically, when dealing with higher merger-split operations or networks with more complex structures (e.g., higher mixing parameters and community density), DCDBFE struggles to maintain its edge over other techniques. These areas are likely to introduce more noise and complexity, which can reduce the robustness of its performance.

Instability may also arise from the technique's sensitivity to changes in community density or the network's dynamic properties, as shown in the varying results for the mixing parameters. Strategies to address instability and to solve the instability problem in DCDBFE, improvements can be made by refining its fitness evaluation mechanism, which might be overly sensitive to certain topological changes like splits or mergers in communities.

Another approach is to enhance its adaptability to diverse network structures by incorporating mechanisms that adjust dynamically as the network evolves, allowing it to maintain high performance across different scenarios. Additionally, leveraging hybrid techniques that combine both evolutionary and local optimization techniques could also stabilize performance, ensuring that DCDBFE doesn't falter when faced with highly dynamic or noisy networks.

Overall, DCDBFE is one of the top-performing methods, especially in phases with moderate changes in network structure. However, its performance declines in cases of extreme dynamics or instability, as seen with larger merger-split operations and high mixing parameters. This suggests the need for refinement in handling more chaotic or dynamic environments to reduce the instability in its performance. This analysis shows that while DCDBFE excels in most scenarios, further tuning is necessary to handle instability, especially in synthetic networks with complex topological changes.

5.5.4 Validation of Results in Phase 4 on Real-World Networks

This section validates the contribution of the proposed technique by comparing it to existing techniques through analysis of various real-world networks. Table 5.4 presents the results of the Friedman test for different real-world networks, including workplace contact

(WC), primary school (PS), high school 2011 (HS2011), high school 2012 (HS2012), cumulative coauthor (CC), and non-cumulative coauthor (NCC) networks. The performance of different techniques was compared using the NMI and ARI metrics. The results demonstrated improvements across real-world networks, which were validated through the Friedman test illustrated in Table 5.4. These results indicate statistically significant differences across all datasets, with the DCDBFE consistently ranked as one of the top-performing techniques in comparison to others.

Table 5.4 Performance validation of DCDBFE using Friedman test on various real-world network datasets in Phase 4

Test dataset	Evaluation metric	Friedman Test Statistic	Friedman p-value	Description
Workplace Contact	NMI	12.79	2.55×10^{-2}	Have significant difference
	ARI	34.57	2.46×10^{-6}	
Primary School	NMI	26.09	8.55×10^{-5}	
	ARI	23.42	2.79×10^{-4}	
High School 2011	NMI	20.00	4.99×10^{-4}	
	ARI	16.91	2.01×10^{-3}	
High School 2012	NMI	21.10	3.02×10^{-4}	
	ARI	19.8	5.47×10^{-4}	
Cumulative Coauthor	NMI	54.42	4.29×10^{-11}	
	ARI	64.51	3.26×10^{-13}	
Non-Cumulative Coauthor	NMI	59.78	3.24×10^{-12}	
	ARI	66.37	1.32×10^{-13}	

The instability observed in more dynamic datasets, such as high school networks, suggests that DCDBFE could benefit from further optimization for networks where relationships and community structures change frequently. Several strategies could address incorporation of dynamics by accounting for temporal changes in community structure, where DCDBFE could become more adaptable to networks like high school interactions, where the community landscape evolves rapidly.

Addressing instability and to solve the instability problem, DCDBFE could benefit from the integration of temporal dynamics within its detection technique. This would allow the technique to track and adjust to evolving community structures rather than assuming static relationships. Additionally, adopting a more adaptive technique that can adjust their parameters based on the network's evolving nature could help improve the technique's stability. For example, integrating local optimization strategies or multi-objective optimization may enable DCDBFE to perform better in environments where the community structure is fluid.

Another strategy could involve fine-tuning its fitness evaluation criteria to be more sensitive to short-term changes in community structures, allowing it to respond more effectively to networks where relationships shift frequently, as seen in social networks like high schools. Overall, DCDBFE has been proven to be one of the top-performing techniques in stable, structured real-world networks, such as workplace contacts and co-author networks. However, it showed some instability in more dynamic environments like high schools, suggesting the need for enhancements to improve its stability in detecting communities in rapidly evolving social networks. By addressing these instability issues, DCDBFE could become more reliable in networks with varying levels of complexity and dynamism, ensuring stable performance across all types of real-world scenarios.

5.6 Summary

In conclusion, Chapter 5 presents the results and discussions on the preliminary study and the analysis of the proposed technique. The findings revealed that the RA similiarity measure outperformed other similarity measures regarding vertex and module attraction.

These techniques demonstrated superior performance in capturing communities structure over time perfectly. The results validate the effectiveness of the proposed technique and its potential for advancing network analysis and understanding dynamics community detection inspired by natural law bio-system.

Overall, the research highlights the significance of applying different similarity measures to enhance their stability in capturing the community networks. The combination of RA similarity with the third level of connectivity stands out as reliable and promising calculation of module attraction. This offers valuable tools for various real-world applications, including social network analysis, disease spread modeling, and communication network design.

The comprehensive analysis presented in Chapter 5 establishes the superiority of DCDBFE compared to existing technique and contributes to the advancement of network analysis methodologies. These findings offer researchers and practitioners new insights and tools to understand dynamics in complex networks more effectively. Additionally, future work could explore the performance on even larger and more complex networks to determine its scalability limits.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS FOR FUTURE RESEARCH

6.1 Introduction

This chapter summarises the overall findings from the current study, discusses the contribution, practical implications and beneficiaries, limitation, and provides suggestions for future work.

6.2 Thesis Summary

This research aimed to develop an enhanced incremental dynamic community detection to improve the stability of community structures in network analysis. Firstly, a similarity analysis for selecting the best similarity measure was proposed as a new calculation for the initial phase of vertex attraction. This new approach involved enhancing the similarity measure index using the resource allocation measure. Beyond the regular selection of a similarity measure, the uncertainty of vertex attraction in forming modules or communities was integrated using module analysis, where module attraction was calculated up to the third level of vertex neighbors. These two approaches were vital due to the importance of detecting communities in the network structure so that critical similarities associated with specific risks could be identified and overfitting could be reduced. The third level of connectivity counting for vertex neighbors was also adopted to identify the most influential vertex for forming a module. The overall experimental evaluation suggests that the similarity analysis using the resource allocation index offers outstanding performance in terms of time efficiency and the quality of the communities detected. Furthermore, the

selection of an appropriate similarity measure demonstrated superiority in the overall community structure compared to other similarity measures. Thus, the findings achieved the first objective of identifying an appropriate similarity measure and module attraction for the community detection.

Secondly, an enhanced incremental dynamic community detection inspired by the bird flock effect, called DCDBFE, was proposed. The concept of birds flocking together was incorporated into the framework of this research. To naturally display community patterns in dynamic networks, this technique simulates the bird flock effect. The design of this technique imitates the three basic rules of bird flocking: separation, alignment, and cohesion. DCDBFE successfully identified high-quality community structures in dynamic networks without setting any parameters. The efficiency of different dynamic community detection in network structures was investigated, with the enhancement of selecting an appropriate similarity measure and module attraction of third-level connections of neighbors. Based on the experimental results, module attraction in the first, second, and third levels of vertex neighbors showed good performance, with the third level performing the best. The choice of the third-level module attraction is not only supported by statistical evaluation but also consistent with theoretical expectations about the mesoscopic scale of communities. Network cohesion typically manifests within 3 level connections or hops radius, making the third-level neighbourhood a structurally meaningful boundary for capturing true module attraction. In short, the findings addressed the second objective of this study, which was to develop an enhanced dynamic community detection based on the bird flock effect using an incremental approach.

Finally, the efficiency of the DCDBFE was tested against several well-known dynamic community detection on the extended LFR benchmark and four real-world networks. Based on the evaluation in the experiment, DCDBFE performed well in finding groups or communities in various types of dynamic networks using common criteria such as the normalized mutual index (NMI) and the adjusted rand index (ARI). These results confirmed that the trade-off between accuracy and stability in dynamic community detection was resolved by maintaining both aspects, thus achieving the third objective.

The entire framework contributed to the knowledge of network structures for dynamic community detection in the literature. The findings in this thesis reaffirm the superiority of the enhanced dynamic community detection framework based on an incremental approach inspired by the bird flock effect for network structures. The reported results demonstrate that selecting an appropriate similarity measure and module attraction of third-layer neighbors provides better performance in network structures compared to traditional techniques for model building.

6.3 Research Contributions

The main aim of this thesis was to develop an enhanced incremental dynamic community detection to improve the identification and stability of community structure in network analysis. The three main contributions of this research are as follows:

1) **Contribution 1: Identification of an appropriate similarity measure and module attraction for community detection technique**.

The appropriate similarity measure for community detection and module attraction for community detection started with the data network preprocessing. The preliminary

analysis showed that the resource allocation index outperformed other methods. Hence, this contribution was further improved by focusing on the vertex distance calculation. The effectiveness of this contribution was evidenced by improved performance metrics demonstrated in Figures 5.1 to 5.9. This achievement ensures that the foundation of the community detection process is optimized, addressing the objective of enhancing network preprocessing for improved detection stability.

2) **Contribution 2: Development of an enhanced dynamic community detection based on incremental approach**.

To address the network structure problem, this study shows that the DCDBFE technique, which uses bird flock effect behavior derived from natural laws, finds stable modules or communities. Experimental evaluations revealed that the proposed technique outperformed existing techniques in both efficiency and stability across different network configurations. Figures 5.10 to 5.21 illustrate these results. This contribution fulfills the objective of designing a novel incremental dynamic detection method that maintains stability and accuracy over time.

3) **Contribution 3: Comprehensive evaluation of the DCDBFE technique by comparison with existing well-known techniques using non-parametric statistical approach such as Friedman test.**

The DCDBFE does not require initial parameter settings. Extensive experiments were carried out on seventeen groups of synthetic networks and four real-world networks to verify the performance of the proposed technique. In terms of stability, the DCDBFE technique consistently demonstrated very good results in quality community detection, as proven in Figures 5.22 to 5.27 for performance evaluation.

6.4 Practical Implications and Beneficiaries

With respect to this section on practical implications, the term graph analytics has emerged, referring to the study and analysis of data that can be transformed into a graph representation. Graph analytics, also known as network analytics, is utilized in various multi-disciplinary and high-impact applications to uncover patterns in real-world systems.

The focus of this work on enhancing incremental dynamic community detection presents a significant advancement in network analysis, particularly addressing the challenge of maintaining stability in dynamic networks. This contribution is directly aligned with several Sustainable Development Goals (SDGs):

1. **Goal 9 (Industry, Innovation, and Infrastructure)**: The development of robust computational tools for managing evolving data in dynamic systems (e.g., telecommunications, transport, and energy grids) enhances infrastructure stability and fosters innovation, especially in real-time community detection for industries like smart cities and Internet of Things (IoT) applications.

2. **Goal 11 (Sustainable Cities and Communities)**: Incremental dynamic community detection optimizes urban mobility, energy distribution, and resource management in smart cities, enabling planners to better predict and adapt to network changes. It also helps in identifying stable community structures in social, transportation, and communication networks, contributing to sustainable urban development.

3. **Goal 13 (Climate Action)**: This method improves the monitoring of environmental networks, enabling timely detection of shifts in ecosystems and weather patterns, thus informing climate adaptation strategies. The stability of dynamic community

detection facilitates faster, real-time responses to emerging climate data, supporting better decision-making for environmental challenges.

4. **Goal 17 (Partnerships for the Goals)**: By strengthening global networks and providing stable community detection methods, this work enhances international collaboration and data-driven decision-making, which are critical for the successful implementation of SDGs.

6.5 Limitations and Recommendations for Future Research Work

Although the proposed DCDBFE technique demonstrates strong performance on synthetic networks, its results on real-world networks are comparatively less optimal. Real networks are often noisy, heterogeneous, and dynamically evolving, which challenges the assumptions of sparsity and modularity inherent in the current method. Gradual changes, overlapping communities, and irregular connectivity can lead to over-segmentation or missed community structures, limiting detection accuracy and stability. Additionally, the current implementation assigns each vertex to a single module, producing non-overlapping communities that may not reflect the complex participation of vertices in multiple groups.

The technique also faces scalability and real-time application limitations, particularly for extremely large networks or live data streams. While DCDBFE improves computational efficiency, processing millions of vertices and edges or handling streaming social media or IoT data requires further optimization. Moreover, domain-specific adaptation is necessary, as different network types; such as social, biological, or technological that exhibit unique structural patterns that may reduce detection quality without fine-tuning.

Future research should address these limitations by adopting the Resource Allocation (RA) index as the similarity measure, which was proven effective in capturing vertex relationships, and by incorporating the third-level connection in module attraction to detect subtle or indirect community structures. Extending DCDBFE to support overlapping or soft membership communities will better represent real networks. Furthermore, enhancing scalability, real-time performance, and domain-specific tuning, as well as testing on dense, heterogeneous, and multilayer networks, will strengthen the technique's robustness, accuracy, and applicability in practical dynamic network scenarios.

REFERENCES

Adamic, L.A., and Adar, E., 2003. Friends and neighbors on the Web. *Social Networks*, 25(3), pp.211–230.

Agarwal, P., Verma, R., Agarwal, A., and Chakraborty, T., 2018. DyPerm: Maximizing permanence for dynamic community detection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10937 LNAI, pp.437–449.

Akbar, S., and Saritha, S.K., 2020. Towards quantum computing based community detection. *Computer Science Review*, 38(3), p.100313.

Alotaibi, N., and Rhouma, D., 2022. A review on community structures detection in time evolving social networks. *Journal of King Saud University - Computer and Information Sciences*, 34(8), pp.5646–5662.

Aynaud, T., Fleury, E., Guillaume, J.-L., and Wang, Q., 2013. Communities in Evolving Networks: Definitions, Detection, and Analysis Techniques. In *Modeling and Simulation in Science, Engineering and Technology*, pp. 159–200.

Bansal, S., Bhowmick, S., and Paymal, P., 2011. Fast Community Detection for Dynamic Complex Networks. In *Communications in Computer and Information Science (Conf.Paper)*, pp. 196–207.

Beauchamp, G., 2021. Flocking in birds increases annual adult survival in a global analysis. *Oecologia*, 197(2), pp.387–394.

Bellaachia, A., and Bari, A., 2011. SFLOSCAN: A biologically-inspired data mining framework for community identification in dynamic social networks. *IEEE SSCI 2011 - Symposium Series on Computational Intelligence - SIS 2011: 2011 IEEE Symposium*

on Swarm Intelligence, (Ssci), pp.156–163.

Blondel, V.D., Guillaume, J.-L.L., Lambiotte, R., and Lefebvre, E., 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), pp.1–12.

Bouhatem, F., Hadj, A.A. El, Souam, F., and Dafeur, A., 2021. Incremental methods for community detection in both fully and growing dynamic networks. *Acta Universitatis Sapientiae, Informatica*, 13(2), pp.220–250.

Cazabet, R., Boudebza, S., and Rossetti, G., 2021. Evaluating community detection algorithms for progressively evolving graphs Perc, M., (ed.). *Journal of Complex Networks*, 8(6), pp.1–30.

Chinichian, N., Kruschwitz, J.D., Reinhardt, P., Palm, M., Wellan, S.A., Erk, S., Heinz, A., Walter, H., and Veer, I.M., 2023. A fast and intuitive method for calculating dynamic network reconfiguration and node flexibility. *Frontiers in Neuroscience*, 17.

Choudhury, N., 2024. Community-Aware Evolution Similarity for Link Prediction in Dynamic Social Networks. *Mathematics*, 12(2).

Christopoulos, K., Baltsou, G., and Tsichlas, K., 2023. Local Community Detection in Graph Streams with Anchors. *Information*, 14(6), p.332.

Christopoulos, K., and Tsichlas, K., 2022. State-of-the-Art in Community Detection in Temporal Networks. In *IFIP Advances in Information and Communication Technology*, pp. 370–381.

Chunaev, P., 2020. Community detection in node-attributed social networks: A survey. *Computer Science Review*, 37, p.100286.

Dai, J., Wang, B., Sheng, J., Sun, Z., Khawaja, F.R., Ullah, A., Dejene, D.A., and Duan, G.,

2019. Identifying Influential Nodes in Complex Networks Based on Local Neighbor Contribution. *IEEE Access*, 7, pp.131719–131731.

Dakiche, N., Benbouzid-Si Tayeb, F., Slimani, Y., and Benatchba, K., 2019. Tracking community evolution in social networks: A survey. *Information Processing & Management*, 56(3), pp.1084–1102.

Danon, L., Díaz-Guilera, A., Duch, J., and Arenas, A., 2005. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09), pp.P09008–P09008.

Elhishi, S., Abu-Elkheir, M., and Abou Elfetouh, A., 2019. Perspectives on the evolution of online communities. *Behaviour & Information Technology*, 38(6), pp.592–608.

Folino, F., and Pizzuti, C., 2014. An Evolutionary Multiobjective Approach for Community Discovery in Dynamic Networks. *IEEE Transactions on Knowledge and Data Engineering*, 26(8), pp.1838–1852.

Fortunato, S., 2010. Community detection in graphs. *Physics Reports*, 486(3–5), pp.75–174.

Ghawi, R., and Pfeffer, J., 2022. A community matching based approach to measuring layer similarity in multilayer networks. *Social Networks*, 68(April 2021), pp.1–14.

Giatsoglou, M., and Vakali, A., 2013. Capturing Social Data Evolution Using Graph Clustering. *IEEE Internet Computing*, 17(1), pp.74–79.

Greene, D., Doyle, D., and Cunningham, P., 2010. Tracking the Evolution of Communities in Dynamic Social Networks. *2010 International Conference on Advances in Social Networks Analysis and Mining*, 176–183 August 2010. IEEE.

Gulbahce, N., and Lehmann, S., 2008. The art of community detection. *BioEssays*, 30(10), pp.934–938.

Hairol Anuar, S.H., Abas, Z.A., Mukhtar, M.F., and Miswan, N.H., 2024. Community Detection in Practice: A Review of Real-World Applications Across Six Themes. *International Journal of Academic Research in Business and Social Sciences*, 14(10), pp.953–996.

Hartmann, T., Kappes, A., and Wagner, D., 2016. Clustering Evolving Networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 280–329.

Holme, P., and Saramäki, J., 2011. Temporal networks. *Physics Reports*, 519(3), pp.97–125.

Holme, P., and Saramäki, J., 2019. *Temporal Network Theory*,

Hubert, L., and Arabie, P., 1985. Comparing partitions. *Journal of Classification*, 2(1), pp.193–218.

Jaccard, P., 1901. Étude comparative de la distribution florale dans une portion des Alpes et du Jura. , (June).

Javed, M.A., Younis, M.S., Latif, S., Qadir, J., and Baig, A., 2018. Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications*, 108(January), pp.87–111.

Jin, D., Yu, Z., Jiao, P., Pan, S., He, D., Wu, J., Yu, P., and Zhang, W., 2021. A Survey of Community Detection Approaches: From Statistical Modeling to Deep Learning. *IEEE Transactions on Knowledge and Data Engineering*, pp.1–1.

Joshi, D., and Patalia, T., 2020. Community Detection Methods and Tools for Various Complex Network. *International Journal of Engineering Research and Technology*, 13(6), pp.1386–1390.

Kadkhoda Mohammadmosaferi, K., and Naderi, H., 2020. Evolution of communities in

dynamic social networks: An efficient map-based approach. *Expert Systems with Applications*, 147, p.113221.

Karatas, A., and Sahin, S., 2018. Application Areas of Community Detection: A Review. *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, 65–70 December 2018. IEEE.

Karatas, A., and Sahin, S., 2022. A Novel Efficient Method for Tracking Evolution of Communities in Dynamic Networks. *IEEE Access*, 10, pp.46276–46290.

Khawaja, F.R., Zhang, Z., Memon, Y., and Ullah, A., 2024. Exploring community detection methods and their diverse applications in complex networks: a comprehensive review. *Social Network Analysis and Mining*, 14(1), p.115.

Lancichinetti, A., and Fortunato, S., 2009. Community detection algorithms: A comparative analysis. *Physical Review E*, 80(5), p.056117.

Lancichinetti, A., Fortunato, S., and Radicchi, F., 2008. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4), p.046110.

Lei, C., Xiao, Y., Jin, S., Huang, T., Zhang, C., and Cheng, M., 2025. A Community Detection Model Based on Dynamic Propagation-Aware Multi-Hop Feature Aggregation. , pp.1–21.

Leicht, E.A., Holme, P., and Newman, M.E.J., 2006. Vertex similarity in networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 73(2).

Liben-Nowell, D., and Kleinberg, J., 2003. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7), pp.1019–1031.

Lijcklama à Nijeholt, D., 2020. Control for Cooperative Autonomous Driving Inspired by

Bird Flocking Behavior.

Lin, Y.-R., Chi, Y., Zhu, S., Sundaram, H., and Tseng, B.L., 2009. Analyzing communities and their evolutions in dynamic social networks. *ACM Transactions on Knowledge Discovery from Data*, 3(2), pp.1–31.

Liu, F., Wu, J., Xue, S., Zhou, C., Yang, J., and Sheng, Q., 2020. Detecting the evolving community structure in dynamic social networks. *World Wide Web*, 23(2), pp.715–733.

Liu, J., Du, Y.-J., Li, Q., and Fu, C.-L., 2018. Social community evolution by combining gravitational relationship with community structure. *Intelligent Data Analysis*, 22(5), pp.1143–1161.

Liu, Q., Zhu, S., Chen, M., and Liu, W., 2022. Detecting Dynamic Communities in Vehicle Movements Using Ant Colony Optimization. *Applied Sciences*, 12(15), p.7608.

Liu, X., Du, Y., Jiang, M., and Zeng, X., 2020. Multiobjective Particle Swarm Optimization Based on Network Embedding for Complex Network Community Detection. *IEEE Transactions on Computational Social Systems*, 7(2), pp.437–449.

Liu, X., and Qiu, L., 2019. Bird Flocking Inspired Control Strategy for Multi-UAV Collective Motion. , (1), pp.1–7.

Nakos, G., 2024. *Elementary Linear Algebra with Applications*, USA: De Gruyter.

Naoki Masuda, and Renaud Lambiotte, 2021. *A Guide To Temporal Networks-World Scientific (2016)*,

Nguyen, N.P., Dinh, T.N., Xuan, Y., and Thai, M.T., 2011. Adaptive algorithms for detecting community structure in dynamic social networks. *Proceedings - IEEE INFOCOM*, pp.2282–2290.

Nooribakhsh, M., Fernández-Diego, M., González-Ladrón-De-Guevara, F., and Mollamotalebi, M., 2024. *Community detection in social networks using machine learning: a systematic mapping study*, Springer London.

Panizo-LLedot, A., Bello-Orgaz, G., and Camacho, D., 2020. A Multi-Objective Genetic Algorithm for detecting dynamic communities using a local search driven immigrant's scheme. *Future Generation Computer Systems*, 110(xxxx), pp.960–975.

Papadopoulos, S., Kompatsiaris, Y., Vakali, A., and Spyridonos, P., 2012. Community detection in Social Media. *Data Mining and Knowledge Discovery*, 24(3), pp.515–554.

Pereira, L.R., Lopes, R.J., and Louçã, J., 2021. Community identity in a temporal network: A taxonomy proposal. *Ecological Complexity*, 45(December 2020), p.100904.

Pranckutė, R., 2021. Scopus and Web of Science stands out for systematic reviews, offering comprehensive coverage across disciplines, including journals, conferences, and patents. *Publications*, 9(1), pp.1–59.

Qu, S., Du, Y., Zhu, M., Yuan, G., Wang, J., Zhang, Y., and Duan, X., 2022. Dynamic Community Detection Based on Evolutionary DeepWalk. *Applied Sciences*, 12(22), p.11464.

Raghavan, U.N., Albert, R., and Kumara, S., 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3), p.036106.

Rahimi, S., Abdollahpouri, A., and Moradi, P., 2018. A multi-objective particle swarm optimization algorithm for community detection in complex networks. *Swarm and Evolutionary Computation*, 39, pp.297–309.

Rand, W.M., 1971. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336), p.846.

Reynolds, C.W., 1987. Flocks-Hers-and-Schools. *Computer Graphics*, 21(4), pp.25–34.

Rossetti, G., 2015. *Social Network Dynamics*.

Rossetti, G., and Cazabet, R., 2019. Community Discovery in Dynamic Networks: A Survey. *ACM Computing Surveys*, 51(2), pp.1–37.

Rossetti, G., Milli, L., Rinzivillo, S., Sîrbu, A., Pedreschi, D., and Giannotti, F., 2018. NDlib: a python library to model and analyze diffusion processes over complex networks. *International Journal of Data Science and Analytics*, 5(1), pp.61–79.

Rossetti, G., Pedreschi, D., and Giannotti, F., 2017. Node-centric Community Discovery: From static to dynamic social network analysis. *Online Social Networks and Media*, 3–4, pp.32–48.

Rosvall, M., and Bergstrom, C.T., 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4), pp.1118–1123.

Salton, G., Wong, A., and Yang, C.S., 1975. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11), pp.613–620.

Seifikar, M., and Farzi, S., 2019. A comprehensive study of online event tracking algorithms in social networks. *Journal of Information Science*, 45(2), pp.156–168.

Shang, J., Li, Y., Sun, Y., Li, F., Zhang, Y., and Liu, J., 2020. MOPIO: A Multi-Objective Pigeon-Inspired Optimization Algorithm for Community Detection. *Symmetry*, 13(1), p.49.

Shao, J., Han, Z., and Yang, Q., 2014. Community Detection via Local Dynamic Interaction.

Shen, X., Yao, X., Tu, H., and Gong, D., 2022. Parallel multi-objective evolutionary optimization based dynamic community detection in software ecosystem. *Knowledge-*

Based Systems, 252, p.109404.

Shetty, R.D., Rashmi, M., Shetty, K.R., and Manoj, T., 2025. Enhanced complex network influential node detection through the integration of entropy and degree metrics with node distance. , pp.1–15.

Sorensen, T., 1948. *A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content*, Munksgaard.

Su, X., et al., 2024. A Comprehensive Survey on Community Detection With Deep Learning. *IEEE Transactions on Neural Networks and Learning Systems*, (Xx), pp.1–21.

Su, X., Cheng, J., Yang, H., Leng, M., Zhang, W., and Chen, X., 2020. IncNSA: Detecting communities incrementally from time-evolving networks based on node similarity. *International Journal of Modern Physics C*, 31(07), p.2050094.

Sun, Y., Sun, Z., Chang, X., Pan, Z., and Luo, L., 2022. Community Detection Based on Fish School Effect. *IEEE Access*, 10, pp.48523–48538.

Sun, Z., Sheng, J., Wang, B., Ullah, A., and Khawaja, F., 2020. Identifying Communities in Dynamic Networks Using Information Dynamics. *Entropy*, 22(4), p.425.

Sun, Z., Sun, Y., Chang, X., Wang, F., Pan, Z., Wang, G., and Liu, J., 2022. Dynamic community detection based on the Matthew effect. *Physica A: Statistical Mechanics and its Applications*, 597(4), p.127315.

Sun, Z., Sun, Y., Chang, X., Wang, Q., Yan, X., Pan, Z., and Li, Z. ping, 2020. Community detection based on the Matthew effect. *Knowledge-Based Systems*, 205.

Taha, K., 2020. Static and Dynamic Community Detection Methods That Optimize a Specific Objective Function: A Survey and Experimental Evaluation. *IEEE Access*,

8(1), pp.98330–98358.

Traag, V.A., Waltman, L., and van Eck, N.J., 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1), pp.1–12.

Vinh, N.X., Epps, J., and Bailey, J., 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11, pp.2837–2854.

Zarayeneh, N., and Kalyanaraman, A., 2021. Delta-Screening: A Fast and Efficient Technique to Update Communities in Dynamic Graphs. *IEEE Transactions on Network Science and Engineering*, 8(2), pp.1614–1629.

Zhou, T., Lü, L., and Zhang, Y.-C., 2009. Predicting missing links via local information. *The European Physical Journal B*, 71(4), pp.623–630.

Zhuang, D., Chang, M.J., and Li, M., 2019. DynaMo: Dynamic Community Detection by Incrementally Maximizing Modularity. *IEEE Transactions on Knowledge and Data Engineering*, 171(March 2020), pp.1–1.

APPENDICES

Appendix A

Coding for DCDBFE

```
import networkx as nx
from sklearn.metrics.cluster import normalized_mutual_info_score
from sklearn.metrics.cluster import adjusted_rand_score
from collections import defaultdict
import time
import datetime
from sklearn import metrics
import math
import matplotlib.pyplot as plt
import numpy as np
from numpy import linalg as LA
import openpyxl

start_time = time.time()
def str_to_int(x):
    return [[int(v) for v in line.split()] for line in x]


def node_addition(G, addnodes, communitys):
    change_comm = set()
    processed_edges = set()

    for u in addnodes:
        neighbors_u = G.neighbors(u)
        neig_comm = set()
        pc = set()
        for v in neighbors_u:
            if v in communitys:
                neig_comm.add(communitys[v])
            pc.add((u, v))
            pc.add((v, u))
        if len(neig_comm) > 1:
            change_comm = change_comm | neig_comm
            lab = max(communitys.values()) + 1
            communitys.setdefault(u, lab)
            change_comm.add(lab)
        else:
            if len(neig_comm) == 1:
```

```
                communitys.setdefault(u, list(neig_comm)[0])
                processed_edges = processed_edges | pc
            else:
                communitys.setdefault(u, max(communitys.values()) + 1)
    return change_comm, processed_edges, communitys


def node_deletion(G, delnodes, communitys):
    change_comm = set()
    processed_edges = set()
    for u in delnodes:
        neighbors_u = G.neighbors(u)
        neig_comm = set()
        for v in neighbors_u:
            if v in communitys:
                neig_comm.add(communitys[v])
            processed_edges.add((u, v))
            processed_edges.add((v, u))
        del communitys[u]
        change_comm = change_comm | neig_comm
    return change_comm, processed_edges, communitys


def edge_addition(addedges, communitys):
    change_comm = set()
    #    print addedges
    #    print communitys
    for item in addedges:
        neig_comm = set()
        neig_comm.add(communitys[item[0]])
        neig_comm.add(communitys[item[1]])
        if len(neig_comm) > 1:
            change_comm = change_comm | neig_comm
    return change_comm


def edge_deletion(deledges, communitys):
    change_comm = set()
    for item in deledges:
        neig_comm = set()
        neig_comm.add(communitys[item[0]])
        neig_comm.add(communitys[item[1]])
        if len(neig_comm) == 1:
            change_comm = change_comm | neig_comm
    return change_comm
```

```
def getchangegraph(all_change_comm, newcomm, Gt):
    Gte = nx.Graph()
    com_key = newcomm.keys()
    for v in Gt.nodes():
        if v not in com_key or newcomm[v] in all_change_comm:
            Gte.add_node(v)
            neig_v = Gt.neighbors(v)
            for u in neig_v:
                if u not in com_key or newcomm[u] in all_change_comm:
                    Gte.add_edge(v, u)
                    Gte.add_node(u)

    return Gte
```

Appendix B

Coding for PHASE 1 Similarity Analysis

```
    # Initialize communities of each node
    # In the first stage, one vertex is one community

nodecount_comm = defaultdict(int)


def CDBFE(G,algorithm_mode=3):
    deg = G.degree()

    # def AA(NA, NB):
    def AA(u, v):

        # comm_nodes = list(NA & NB)
        comm_nodes = list(u & v)
        #print("ini adalah node :", comm_nodes)
        sim = 0
        for node in comm_nodes:
            degnode = deg[node]
            if deg[node] == 1:
                degnode = 1.1
            # sim = sim + (1.0 / math.log(degnode))  ##AA similarity measure
            sim = sim + (1.0 / (degnode))       ##RA similarity measure
        return sim

        # Compute the jaccard similarity coefficient of two vertex
    def simjkd(u, v):
        set_v = v
        set_u = u
        jac = len(set_v & set_u) * 1.0 / len(set_v | set_u) # Jaccard
measure
        #jac = len(set_v & set_u) * 1.0 / (len(set_v)*len(set_u))  # Leicht-
Holme-Newman measure
        return jac

    def simsltn(u, v):
        set_v = v
        set_u = u
        salton = len(set_v & set_u) * 1.0 / math.sqrt(len(v) * len(u)) #
Salton measure
```

```
        # salton = len(set_v & set_u) * 1.0 / (len(v)+len(u))  # Sorensen
Similarity measure
        return salton

    #coefficient
    def simjkdcoff(u,v):
        set_v = set(G.neighbors(v))
        set_v.add(v)
        set_u = set(G.neighbors(u))
        set_u.add(u)
        jac = len(set_v & set_u) * 1.0 / len(set_v | set_u)  # index Jaccard
        #jac = len(set_v & set_u) * 1.0 / (len(set_v) *
len(set_u))  ###Leicht-Holme-Newman
        return jac

    def simsltncoff(u, v):
        set_v = set(G.neighbors(v))
        set_v.add(v)
        set_u = set(G.neighbors(u))
        set_u.add(u)
        salton = len(set_v & set_u) * 1.0 / math.sqrt(len(set_v) *
len(set_u))  # index Salton
        # salton = len(set_v & set_u) * 1.0 / (len(set_v) +
len(set_u))  ##Sorensen Similarity
        return salton
    node_community = dict(zip(G.nodes(), G.nodes()))

# Compute the submodules Vertex Attraction
    # In the second stage, the Submodules is calculated
    st = {}  # storge the AA
    # compute the AA
    for node in G.nodes():
        Nv = sorted(G.neighbors(node))
        for u in Nv:
            Nu = G.neighbors(u)
            keys = str(node) + '_' + str(u)
            #st.setdefault(keys, simsltn(set(Nv), set(Nu)))
            #st.setdefault(keys, simjkd(set(Nv), set(Nu)))
            #st.setdefault(keys, AA(set(Nv), set(Nu)))
            if algorithm_mode == 1 :
                st.setdefault(keys, AA(set(Nv), set(Nu)))
            elif algorithm_mode == 2 :
                st.setdefault(keys, simjkd(set(Nv), set(Nu)))
```

```
            elif algorithm_mode == 3 :
                st.setdefault(keys, simsltn(set(Nv), set(Nu)))
            else:
                exit()
    if algorithm_mode == 1:
        print('AA index,done')
    elif algorithm_mode == 2:
        print('JACCARD index,done')
    elif algorithm_mode == 3:
        print('SALTION index,done')
    else:
        pass

    for node in G.nodes():
        # The degree of each node
        deg_node = deg[node]
        flag = True
        maxsimdeg = 0
        selected = node
        if deg_node == 1:
            node_community[node] =
node_community[list(G.neighbors(node))[0]]
        else:
            for neig in G.neighbors(node):
                deg_neig = deg[neig]
                if flag is True and deg_node <= deg_neig:
                    flag = False
                    break

            if flag is False:
                for neig in sorted(G.neighbors(node)):
                    deg_neig = deg[neig]
                    if algoritm_mode == 1:
                        keysaa = str(node) + '_' + str(neig)
                        nodesim = st[keysaa]
                    elif algorithm_mode == 2:
                        #keys = str(node) + '_' + str(neig)
                        nodesim = simjkdcoff(node, neig)
                    elif algorithm_mode == 3:
                        nodesim = simsltncoff(node, neig)
                    else:
                        pass

                    # Compute the node attraction
                    nodesimdeg = deg_neig * nodesim
```

```
if nodesimdeg > maxsimdeg:
    selected = neig
    maxsimdeg = nodesimdeg
node_community[node] = node_community[selected]
```

Appendix C

Coding for PHASE 2 Module Analysis

```
# The internal degree of node v in a community
def per(G, v, node_community):
    neiglist1 = G.neighbors(v)  # First-layer neighbors
    in_v = 0
    second_layer_in_v = 0  # Second-layer connectivity
    third_layer_in_v = 0  # Third-layer connectivity
    fourth_layer_in_v = 0  # Fourth-layer connectivity
    global nodecount_comm

    # First-layer connectivity
    for neig in neiglist1:
        if node_community[neig] == node_community[v]:
            in_v += 1
        else:
            nodecount_comm[node_community[neig]] += 1

    # Second-layer connectivity
    for neig in neiglist1:
        neiglist2 = G.neighbors(neig)  # Neighbors of second layer
        for neig2 in neiglist2:
            if neig2 != v and node_community[neig2] == node_community[v]:
                second_layer_in_v += 1

     # Third-layer connectivity
        neiglist3 = G.neighbors(neig2)  # Neighbors of third layer
        for neig3 in neiglist3:
                if neig3 != neig and node_community[neig3] ==
node_community[v]:
                    third_layer_in_v += 1

     # Fourth-layer connectivity
        neiglist4 = G.neighbors(neig3)  # Neighbors of fourth layer
        for neig4 in neiglist4:
                    if neig4 != neig2 and node_community[neig4] ==
node_community[v]:
                        fourth_layer_in_v += 1
    cin_v = 1.0 * (in_v * in_v) + second_layer_in_v + third_layer_in_v +
fourth_layer_in_v  # Include fourth-layer connectivity
    per = cin_v
    return per
```

Appendix D

Coding for PHASE 3 Bird Flock Effect

```
## Simulate the Bird Flock Effect
    old_persum = -(2 ** 63 - 1)
    old_netw_per = -(2 ** 63 - 1)

    persum = old_persum + 1
    netw_per = old_netw_per + 0.1
    maxit = 20
    itern = 0

    print("loop begin:")
    while itern < maxit:
        itern += 1
        old_netw_per = netw_per
        old_persum = persum
        persum = 0
        for node in G.nodes():
            neiglist = sorted(G.neighbors(node))
            cur_p = per(G, node, node_community)  #
            nodeneig_comm = nodecount_comm.keys()
            cur_p_neig = 0

            for neig in neiglist:
                cur_p_neig += per(G, neig, node_community)
            try :
                for neig_comm in nodeneig_comm:

                    node_pre_comm = node_community[node]
                    new_p_neig = 0
                    if node_pre_comm != neig_comm:
                        try:
                            node_community[node] = neig_comm
                            new_p = per(G, node, node_community)

                            if cur_p <= new_p:

                                if cur_p == new_p:
                                    for newneig in neiglist:
                                        new_p_neig += per(G, newneig,
node_community)
```

```
                                if cur_p_neig < new_p_neig:
                                    cur_p = new_p
                                    cur_p_neig = new_p_neig
                                else:
                                    node_community[node] = node_pre_comm

                            else:
                                for newneig in neiglist:
                                    new_p_neig += per(G, newneig,
node_community)
                                cur_p = new_p
                                cur_p_neig = new_p_neig
                        else:
                            node_community[node] = node_pre_comm
                    except:
                        pass
            except:
                print('terjadi error')
            persum += cur_p
    print("loop done")
    return node_community
```

Appendix E

Coding for PHASE 4 Evaluate DCDBFE

```
def Errorrate(clusters, classes, n):
    A = np.zeros((n, len(clusters)), int)
    C = np.zeros((n, len(classes)), int)
    k = 0
    for nodelist in clusters:
        for node in nodelist:
            A[node - 1][k] = 1
        k = k + 1
    k = 0
    for nodelist in classes:
        for node in nodelist:
            C[node - 1][k] = 1
        k = k + 1
    t = A.dot(A.T) - C.dot(C.T)
    errors = LA.norm(t)
    return errors
```



```
def purity_score(clusters, classes):
    A = np.c_[(clusters, classes)]
    n_accurate = 0.

    for j in np.unique(A[:, 0]):
        z = A[A[:, 0] == j, 1]
        x = np.argmax(np.bincount(z))
        n_accurate += len(z[z == x])
    return n_accurate / A.shape[0]


def conver_comm_to_lab(comm1):
    overl_community = {}
    for node_v, com_lab in comm1.items():
        if com_lab in overl_community.keys():
            overl_community[com_lab].append(node_v)
        else:
            overl_community.update({com_lab: [node_v]})
    return overl_community
```

```
def getscore(comm_true, comm_dete, num):
    actual = []
    baseline = []
    for j in range(len(comm_true)):
        for c in comm_true[j]:
            flag = False
            for k in range(len(comm_dete)):
                if c in comm_dete[k] and flag == False:
                    flag = True
                    actual.append(j)
                    baseline.append(k)
                    break

    NMI1 = metrics.normalized_mutual_info_score(actual, baseline)
    ARI1 = metrics.adjusted_rand_score(actual, baseline)
    Purity1 = purity_score(baseline, actual)
    errors=Errorrate(comm_dete,comm_true,num)
    # errors = 0
    print('nmi', NMI1)
    print('ari', ARI1)
    print('purity', Purity1)
    print('rate error', errors)

    return NMI1, ARI1, Purity1, errors
```



```
def drawcommunity(g, partition, filepath):
    pos = nx.spring_layout(g)
    count1 = 0
    t = 0
    node_color = ['#66CCCC', '#FFCC00', '#99CC33', '#CC6600', '#CCCC66',
'#FF99CC', '#66FFFF', '#66CC66', '#CCFFFF',
                  '#CCCC00', '#CC99CC', '#FFFFCC']

    for com in set(partition.values()):
        count1 = count1 + 1.
        list_nodes = [nodes for nodes in partition.keys()
                      if partition[nodes] == com]

        nx.draw_networkx_nodes(g, pos, list_nodes, node_size=220,
                               node_color=node_color[t])
        nx.draw_networkx_labels(g, pos)
        t = t + 1

    nx.draw_networkx_edges(g, pos, with_labels=True, alpha=0.5)
```

```
    plt.savefig(filepath)
    plt.show()


#################################################################
# ----------main-----------------
edges_added = set()
edges_removed = set()
nodes_added = set()
nodes_removed = set()
G = nx.Graph()
print("1. AA Similarity\n"
      "2. Jaccard Similarity\n"
      "3. Salton Similarity\n")
algoritm_similarity = input("Input name of Similarity Algorithm :")
algoritm_mode = None
if int(algoritm_similarity) == 1:
    algoritm_mode = 1
elif int(algoritm_similarity) == 2:
    algoritm_mode = 2
elif int(algoritm_similarity) == 3:
    algoritm_mode = 3
else:
    exit()

#allpath = './data/trydataset/files2.txt'
# allpath = './input/files.txt'
allpath = './input/merge40.txt'
with open(allpath, 'r') as f:
    allpathlist = f.readlines()
    f.close()
# allpathlists=allpathlist[0].strip('\n')
pathfile = ''
for pt in allpathlist:
    pathfile = pt.strip('\n')
    print(pathfile)
#    path = './data/trydataset/' + pathfile + '/'
    path = './input/' + pathfile + '/'
    edge_file = ''
    comm_file = ''
    G.clear()
    # read edgefile list, where storage the filename of each snapshot
    edgefilelist = []
    commfilelist = []
    with open(path + 'edgeslist.txt', 'r') as f:
```

```
    edgefilelist = f.readlines()
    f.close()
edge_file = edgefilelist[0].strip('\n')
with open(path + 'commlist.txt', 'r') as f:
    commfilelist = f.readlines()
    f.close()
comm_file = commfilelist[0].strip('\n')

# path='./LFR/t/'
# path='./data/test/'
with open(path + edge_file, 'r') as f:

    edge_list = f.readlines()
    for edge in edge_list:
        edge = edge.split()
        G.add_node(int(edge[0]))
        G.add_node(int(edge[1]))
        G.add_edge(int(edge[0]), int(edge[1]))
    f.close()
G = G.to_undirected()

nodenumber = G.number_of_nodes()
with open(path + comm_file, 'r') as f:
    comm_list = f.readlines()
    comm_list = str_to_int(comm_list)  # 真实社区
    f.close()
```



```
# The third stage is to simulate the Bird Flock Effect

comm = {}
comm = CDBFE(G,algoritm_mode)

initcomm = conver_comm_to_lab(comm)
comm_va = list(initcomm.values())
commu_num = len(comm_va)
tru_num = len(comm_list)
NMI, ARI, Purity, Errors = getscore(comm_list, comm_va, nodenumber)
import xlsxwriter, os
if int(algoritm_similarity) == 1:
    file_exists = os.path.exists('result_score_LFR_aa.xlsx')
    if file_exists == True:
        path_score = 'result_score_LFR_aa.xlsx'
    else:
        workbook = xlsxwriter.Workbook('result_score_LFR_aa.xlsx')
```

```
            path_score = 'result_score_LFR_aa.xlsx'
            f = open(path_score,'a+')
            f.write('path'+"\t"+'NMI'+"\t"+'ARI'+"\t"+'Purity'+'\t'+'detecte
d_community_number'+'ture_community_number'+'errors'"\n" )
            f.close()
            wb=openpyxl.Workbook(path_score)
            wb.save(path_score)

    elif int(algoritm_similarity) == 2:

        file_exists = os.path.exists('result_score_LFR_jaccard.xlsx')
        if file_exists == True:
            path_score = 'result_score_LFR_jaccard.xlsx'
        else:
            workbook = xlsxwriter.Workbook('result_score_LFR_jaccard.xlsx')
            path_score = 'result_score_LFR_jaccard.xlsx'
            f = open(path_score,'a+')
            f.write('path'+"\t"+'NMI'+"\t"+'ARI'+"\t"+'Purity'+'\t'+'detecte
d_community_number'+'ture_community_number'+'errors'"\n" )
            f.close()
            wb=openpyxl.Workbook(path_score)
            wb.save(path_score)

    elif int(algoritm_similarity) == 3:

        file_exists = os.path.exists('result_score_LFR_salton.xlsx')
        if file_exists == True:
            path_score = 'result_score_LFR_salton.xlsx'
        else:
            workbook = xlsxwriter.Workbook('result_score_LFR_salton.xlsx')
            path_score = 'result_score_LFR_salton.xlsx'
            f = open(path_score,'a+')
            f.write('path'+"\t"+'NMI'+"\t"+'ARI'+"\t"+'Purity'+'\t'+'detecte
d_community_number'+'ture_community_number'+'errors'"\n" )
            f.close()
            wb=openpyxl.Workbook(path_score)
            wb.save(path_score)
    else:
        exit()
    wb = openpyxl.load_workbook(filename=path_score)
    sheetname = path[:len(path) - 1].split('/')
    fix_pat = sheetname[len(sheetname)-1]
    print(fix_pat)
    ws = wb.create_sheet(fix_pat)
```

```
    row = ['path', 'NMI', 'ARI', 'Purity', 'detected_community_number',
'ture_community_number', 'errors']
    ws.append(row)
    row = ['1', str(NMI), str(ARI), str(Purity), str(commu_num),
str(tru_num), str(Errors)]
    ws.append(row)

    start = time.time()
    G1 = nx.Graph()
    G2 = nx.Graph()
    G1 = G

    l = len(edgefilelist)
    for i in range(1, l):
        print('begin loop:', i)
        comm_new_file = open(path + commfilelist[i].strip('\n'), 'r')
        comm_new = comm_new_file.readlines()
        comm_new_file.close()
        comm_new = str_to_int(comm_new)

        edge_list_new_file = open(path + edgefilelist[i].strip('\n'), 'r')
        edge_list_new = edge_list_new_file.readlines()
        edge_list_new_file.close()

        for line in edge_list_new:
            temp = line.strip().split()
            G2.add_edge(int(temp[0]), int(temp[1]))

        # The total number of vertices in the current time step and the previous time
step, which are related to the two sets
        total_nodes = set(G1.nodes()) | set(G2.nodes())

        nodes_added = set(G2.nodes()) - set(G1.nodes())
        # print (' Increase the vertex set to:: ',nodes_added)
        nodes_removed = set(G1.nodes()) - set(G2.nodes())
        # print (' To delete a vertex set, it is:',nodes_removed)

        edges_added = set(G2.edges()) - set(G1.edges())
        # print (' Increase the edge set as:',edges_added)
        edges_removed = set(G1.edges()) - set(G2.edges())
        # print (' Delete the edge set as: ',edges_removed)

        all_change_comm = set()
```

```
        # Add vertex processing
##############################################################
        addn_ch_comm, addn_pro_edges, addn_commu = node_addition(G2,
nodes_added, comm)
        edges_added = edges_added - addn_pro_edges  # Remove the processed
edges
        all_change_comm = all_change_comm | addn_ch_comm

        # Delete vertex processing
##############################################################
        deln_ch_comm, deln_pro_edges, deln_commu = node_deletion(G1,
nodes_removed, addn_commu)
        all_change_comm = all_change_comm | deln_ch_comm
        edges_removed = edges_removed - deln_pro_edges

        # Add edge processing
##############################################################
        #    print('edges_added',edges_added)
        adde_ch_comm = edge_addition(edges_added, deln_commu)
        all_change_comm = all_change_comm | adde_ch_comm
        #    print('all_change_comm',all_change_comm)

      # Delete edge processing
##############################################################
        dele_ch_comm = edge_deletion(edges_removed, deln_commu)
        all_change_comm = all_change_comm | dele_ch_comm
        #    print('all_change_comm',all_change_comm)
        unchangecomm = ()  # 未改变的社区标签
        newcomm = {}  # 格式为 {节点: 社区}
        newcomm = deln_commu  # Adding and deleting edges is only processed
on existing vertices, no new nodes will be added, and vertices will be
deleted (previously processed)
        unchangecomm = set(newcomm.values()) - all_change_comm
        unchcommunity = {key: value for key, value in newcomm.items() if
value in unchangecomm}
        # Find out the subgraph corresponding to the changed community, and
then use bird flock effect dynamics to find out the new community structure
on the subgraph, and add the unchanged community structure to get the new
community structure.
```

```
        Gtemp = nx.Graph()
        Gtemp = getchangegraph(all_change_comm, newcomm, G2)

        unchagecom_maxlabe = 0
        if len(unchangecomm) > 0:
            unchagecom_maxlabe = max(unchangecomm)

        if Gtemp.number_of_edges() < 1:  # 社区未发生变化
            comm = newcomm
        else:
            try:
                getnewcomm = CDME(Gtemp,algoritm_mode)
                print('===========')
                mergecomm = {}  # The merged word
                mergecomm.update(unchcommunity)
                mergecomm.update(getnewcomm)
                # mergecomm=dict(**unchcommunity, **getnewcomm )
                comm = mergecomm
                detectcom = list(conver_comm_to_lab(comm).values())
                commu_num = len(detectcom)
                tru_num = len(comm_new)
            except:
                pass
            print (detectcom)
        nodenumber = G2.number_of_nodes()

        NMI, ARI, Purity, Errors = getscore(comm_new, detectcom,
nodenumber)  # The first parameter data is the real community, and the
second parameter is the detected community
        G1.clear()
        G1.add_edges_from(G2.edges())
        G2.clear()

        row = [str(i + 1), str(NMI), str(ARI), str(Purity), str(commu_num),
str(tru_num), str(Errors)]
        ws.append(row)
    wb.save(path_score)
print('all done')

print(time.time()-start, "second")
```

Appendix F

Coding for Validation using Statistical Approach

Friedman Test

```
import numpy as np
import pandas as pd
from scipy.stats import friedmanchisquare
import matplotlib.pyplot as plt
from matplotlib import rcParams

# Set Times New Roman as the default font for plots
rcParams['font.family'] = 'Times New Roman'
rcParams['font.size'] = 12  # Adjust font size as needed

# Data for each method over 20 time steps (Purity values from Table 5.2)
data = {
    'Time step': np.arange(1, 21),
    'Jaccard': [1.0, 1.0, 0.9982111951844648, 0.9972211458586918, 1.0, 1.0,
1.0, 1.0, 0.9951584310044893, 0.995681901970465, 1.0, 0.9952944494847594,
1.0, 0.9941213417461153, 1.0, 1.0, 1.0, 0.9504916757341733,
0.9937600558558373, 0.995588275445985],
    'Salton': [1.0, 0.9900863473793481, 0.9824494800291275,
0.9777190063723341, 0.9748947410157666, 0.9744092103657017,
0.9780521606385462, 0.9722811494336026, 0.9731959375591812,
0.9911628273588895, 0.9666380228294732, 0.9836252074798909,
0.996688539030269, 0.9754335649482632, 0.9876382553542464,
0.9554911449213433, 0.9668627400472883, 0.9196285192972731,
0.9335191333594653, 0.9720661436066648],
    'Sorensen': [1.0, 0.9957314783724475, 0.9883028516440615,
0.9846300716905962, 0.9939217749903222, 0.9886996211721061,
0.9892279586146379, 0.9777513646131534, 0.9881461146417934,
0.9911628273588895, 0.9875867620629594, 0.9929840945996107, 1.0,
0.9864165294010817, 1.0, 0.9811740857878382, 0.9965361647048402,
0.9438448544148703, 0.9818330345458103, 0.9880655994622115],
    'LHN': [1.0, 1.0, 0.9982111951844648, 0.9972211458586918, 1.0, 1.0, 1.0,
1.0, 0.9951584310044893, 0.995681901970465, 1.0, 0.9952944494847594, 1.0,
0.9941213417461153, 1.0, 1.0, 1.0, 0.9504916757341733, 0.9937600558558373,
0.995588275445985],
    'AA': [1.0, 1.0, 0.9982111951844648, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 0.9844581012576826, 1.0, 1.0, 1.0, 0.9504916757341733, 1.0,
0.9920610572098854],
```

```
    'RA': [1.0, 1.0, 0.9982111951844648, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 0.9941213417461153, 1.0, 1.0, 1.0, 0.9504916757341733, 1.0,
0.995588275445985]
}

# Convert the data into a pandas DataFrame
df = pd.DataFrame(data)

# Exclude the 'Time step' column for analysis
df_values = df.drop(columns=['Time step'])

# Perform the Friedman test to compare the ranks of the methods across time
steps
friedman_stat, p_value = friedmanchisquare(df_values['Jaccard'],
df_values['Salton'], df_values['Sorensen'], df_values['LHN'],
df_values['AA'], df_values['RA'])
print(f"Friedman test statistic: {friedman_stat}, p-value: {p_value}")
```